

Study **R**eport

MOD-6SR/0178
JANUARY 1978

TM-79567

PAYLOAD OPERATIONS CONTROL CENTER NETWORK (POCCNET) SYSTEMS DEFINITION PHASE

(NASA-TM-79567) PAYLOAD OPERATIONS CONTROL
CENTER NETWORK (POCCNET) SYSTEMS DEFINITION
PHASE STUDY REPORT (NASA) 287 p
HC A13/MF A01

N78-24160

CSCI 22A

Unclass

G3/12 21288



NASA

National Aeronautics and
Space Administration

Goddard Space Flight Center

PAYLOAD OPERATIONS CONTROL CENTER NETWORK
(POCCNET)
SYSTEMS DEFINITION PHASE
STUDY REPORT

JANUARY 1978

PREFACE

PAYLOAD OPERATIONS CONTROL CENTER NETWORK
(POCCNET)

SYSTEMS DEFINITION PHASE

STUDY REPORT

This report presents the results of the studies performed during the systems definition phase of Poccnnet. Work is still being done in a number of study areas; in these cases, the conclusions to the time of this writing are presented.

This report describes the concept of Poccnnet as a system of standard POCCs and also includes an analysis of system requirements and an evaluation of alternative systems concepts. Design of some of the subsystems which will make up Poccnnet has begun, and these preliminary designs are also presented.

In addition, various methods for development of highly reliable, reusable software were evaluated, and a recommended software engineering standard approach for Poccnnet was developed. A number of POCC application areas, such as command management, on-board computer (OBC) support, and simulation, were also studied. Other areas of investigation included the operation of Poccnnet systems, the facility requirements for Poccnnet, and using Poccnnet.

The motivation for the development of the Poccnnet concept was the change in requirements for payload support brought on by standard payloads and launch systems which will become operational in the near future. These systems will lower the cost of developing a payload and placing it in orbit and, at the same time, decrease the time available for development of the payload control center. In response to these changing requirements, GSFC must devise a methodology through which POCCs can be developed at lower cost and with shorter lead time than in the past.

The approach taken initially to solve this problem was to design a hardware/software system to respond to these new requirements building on the experience of the past. This system was envisioned as a network of standard POCCs, interconnected to provide backup capability and access to support and communications facilities.

As the design of this system progressed, it became apparent that a more comprehensive approach was needed and that, to do the job correctly, a new methodology for POCC and POCC support systems development was required. This methodology had

to span hardware, software, and operations with emphasis on modularity and standardization in order to be responsive to ever-changing requirements and to take full advantage of changing technology. Therefore, the study has evolved into a broad-based program within the Mission Operations Division which will work toward achieving standard approaches to payload support at GSFC in the 1980s.

The Poccnet Systems Definition Phase Study was divided into the following seven areas: requirements analysis, systems engineering, host computer systems (the computer systems which will run POCC and Poccnet applications), interprocess communication, applications engineering, operating Poccnet, and using Poccnet.

The main body of this report provides an extensive coverage of the issues and the conclusions reached in each of the study areas. In most cases, the complete and in-depth documentation of the study activities is relegated to the appendixes, which are published under a separate cover. These include documents describing the Systems Test and Operation Language (STOL), the MMS Payload Data Base preliminary design, the Systems Implementation Languages Study, Poccnet protocols, the Software Engineering Standard Approach, the Data Base Storage System, and various aspects of the Inter-Process Communication System.

Many problems were encountered in the study areas of computer systems interconnection, virtual device specifications, and software engineering tools/configuration management data base. These three areas are responsible for the 9-month delay in issuing this report and are still incomplete. The study manager finally decided to publish this work in its present form rather than wait any longer for results in these areas.

The problem in all three of these incomplete study areas is fundamentally the same; that is, it is the conviction of the study manager that it is more important that Poccnet be squarely in the mainstream of computer systems hardware and software technology and international standardization trends than it is to meet the original study schedule. This is because Poccnet is conceived first and foremost as a cost-control device for payload operations at GSFC in the 1980s and not as a short-term solution for any particular payload support problem. Special-purpose hardware and complex and troublesome payload-unique software drive costs up and reliability down and therefore must be reduced to the absolute minimum. This can only be accomplished through the widespread utilization of commercially available, industry-standard computer systems and software, together with the reduction of

payload uniqueness to the level of data base descriptions plus an absolute minimum of additional special software modules.

In each of the incomplete study areas, the Pocnet study has shown that the mainstream of applicable technology is just now emerging clearly and unmistakably.

Pocnet study activities have kept fully cognizant of, and compatible with, the crest of the technology waves in these areas; these waves are just now beginning to break and Pocnet is surfing them, as described in the following paragraphs.

In computer systems interconnection, the under-\$2000 ADCCP industry-standard, high-speed serial channel is just now upon us, and we are currently prototyping the first commercially available implementation for use in Pocnet.

In virtual device specifications, Pocnet is in front of the crest. There are no adequate near-term industry standards; therefore, Pocnet will adapt vendor software which will become commercially available in 1978. In the meantime, Pocnet is fully cognizant of, and intends to be compliant with, international standardization activities in the area of distributed systems interconnection including the virtual device and process control levels. In fact the Pocnet study manager is the leader of the U.S. delegation to the cognizant ISO subcommittee.

Finally, software engineering tools/configuration management data base technology is the area in which the Pocnet study is concentrating most of its resources. As a direct result of continuing emphasis on identifying and selecting mainstream trends, Pocnet has more promise of future participation in the fruits of widespread software commonality and standardization than any project in GSFC history. The Pocnet standard applications processor and standard operating system is the most widely used real-time operating system in history. The standard data base management system is the highest rated system available today and is fully compliant with CODASYL industry standards. The Pocnet software engineer is aggressively promoting the NASA-wide support and eventual adoption of the forthcoming DOD standard real-time programming language. The Pocnet program is participating with other elements at GSFC to standardize a single STOL nucleus and a standard payload data base description language to provide common human-to-machine interfaces for all phases of the payload life cycle. In 1978, Pocnet is developing a prototype standard software engineering workbench/program development library/configuration management data base for use in Pocnet. These software facilities will be evaluated objectively under the aegis of the GSFC Software Engineering Laboratory, with which Pocnet has associated since its inception.

In summary, in the three areas cited (computer systems interconnection, virtual device specifications, and software engineering tools/configuration management data base), the Poccnnet study was delayed until enough technology cards became face up to enable a definitive conclusion to be reached as to where the mainstream would lie in the 1980s. We feel confident that we now have a handle on those areas and, in 1978, the ongoing activities in those areas will lead to procurements squarely in those mainstreams.

Concurrent with the publication of this report, a program plan is being submitted which integrates the results of the definition phase study with specific management initiatives. This plan specifies schedules and projects, as well as further studies still underway. This last round of studies is required to complete the definition of Poccnnet subsystems at a pace and level sufficient to support procurement activities for the Space Telescope Control Center, MSOCC I and II refurbishment (including XDS Sigma 5 replacement, Shuttle Payload Interface Facility), and other procurements related to Poccnnet such as the replacement of the IBM 360-65.

The study manager wishes to thank all those who contributed to the Poccnnet conceptual phase activity and to this systems definition study report. Particular thanks must go first to Jerold Hahn, the Poccnnet computer systems manager, for his far-reaching concepts and strenuous systems acquisition activities in support of Poccnnet. Thanks are also due to Walter Truszkowski, the Poccnnet software engineer, for his work, particularly on software engineering standard approaches; to Robert Adams, the Poccnnet operations engineer, for his development of Poccnnet operational concepts; to Roger Tetrick, the Poccnnet user interface engineer, for supporting the development of Poccnnet requirements; and to Fred Brosi and Edward Anderson of Computer Sciences Corporation for their management of the technical study support and preparation of the reports.

Other major contributors to the Poccnnet study are listed below in sincere appreciation for their help.

NASA Goddard Space Flight Center

William Fortney
Gardiner Hall
Robert Owen
William Slade

Computer Sciences Corporation

Roger Bieri
Dr. Amar Macker
James O'Brien
Timothy Swanson

OA0 Corporation

Preston Burch
Arthur Chomas
Morris Gunzburg
Bradley Johnson
Bruce Larsen
Dr. John Nebb

Consultants

Dr. David Farber
Dr. Peter Freeman
Dr. William Wulf

Westinghouse Electric Corporation

David Carey
Anthony Elenbaas
L. L. King
John Nieberding

University of Maryland

Dr. Ashok Agrawala
Jon Agre
Dr. Victor Basili
Raymond Bryant
James Franklin
Karen Gordon
Dr. Marvin Zelkowitz

Richard desJardins
Study Manager
Mission Operations Division
January 23, 1978

CONTENTS

<u>Paragraph</u>		<u>Page</u>
SECTION 1		
INTRODUCTION		
SECTION 2		
EXECUTIVE SUMMARY		
2.1	General	2-1
2.2	Systems Requirements	2-1
2.2.1	Shuttle Mission Control Center	2-1
2.2.2	NASCOM and STDN	2-2
2.2.3	Kennedy Space Center	2-4
2.2.4	IUS/SSUS Operations Control Center(s)	2-4
2.2.5	Spacelab Payload Operations Control Center	2-5
2.2.6	Operational Support Computing	2-5
2.2.6.1	Orbit Data Operations	2-5
2.2.6.2	Flight Dynamics Computing	2-6
2.2.6.3	Mission Planning and Scheduling	2-6
2.2.7	Data Processing Facilities	2-7
2.2.7.1	Telemetry On-Line Processing System	2-7
2.2.7.2	Image Processing Facility	2-7
2.2.7.3	Spacelab Non-Time-Critical Data Processing	2-7
2.2.8	Data Accountability System	2-8
2.2.9	Mission Operations Center Mission Control Room	2-8
2.2.10	External Experimenters	2-8
2.2.11	Miscellaneous Interfaces	2-9
2.2.12	Model POCC Requirements	2-9
2.3	Systems Engineering	2-10
2.3.1	Systems Engineering Approach	2-10
2.3.1.1	Generic System Drivers	2-10
2.3.1.2	Synthesis of Systems Concept	2-14
2.3.1.3	Quantitative Requirements	2-16
2.3.2	Systems Studies	2-17
2.3.3	Software Engineering Standard Approach	2-18

CONTENTS (Cont)

<u>Paragraph</u>		<u>Page</u>
2.4	Host Computer Systems	2-18
2.4.1	Applications Processors	2-19
2.4.2	Data Base Storage System	2-20
2.4.3	Virtual Interface Processor	2-21
2.4.4	Gateways	2-22
2.5	Inter-Process Communication System	2-23
2.5.1	Inter-Process Communication Requirements Analysis	2-23
2.5.2	Design Alternatives	2-24
2.5.3	IPC System Description	2-24
2.6	Applications Engineering	2-26
2.6.1	DBS Applications	2-26
2.6.2	Simulation	2-27
2.6.3	On-Board Computer Support	2-27
2.6.4	Command Memory Management	2-28
2.7	Operations	2-29
2.8	Using Poccnet	2-29
2.8.1	Obtaining Information About Poccnet	2-30
2.8.2	Formal Requirements	2-30
2.8.3	User Operations	2-30

SECTION 3

SYSTEMS REQUIREMENTS

3.1	General	3-1
3.2	External Systems Interfaces	3-1
3.2.1	Shuttle Mission Control Center	3-2
3.2.1.1	Telemetry	3-2
3.2.1.2	Commanding	3-3
3.2.1.3	Operational Information	3-6
3.2.2	NASCOM/STDN/NCC	3-8
3.2.2.1	Introduction	3-8
3.2.2.2	TDRSS Service Capabilities	3-8

CONTENTS (Cont)

<u>Paragraph</u>		<u>Page</u>
3.2.2.3	Telemetry Data Interfaces	3-10
3.2.2.4	Command Data Interfaces	3-11
3.2.2.5	Network Control Center	3-12
3.2.3	Kennedy Space Center	3-18
3.2.4	IUS/SSUS Operations Control Center	3-24
3.2.5	Spacelab POCC	3-29
3.2.6	Operational Support Computing	3-29
3.2.6.1	Orbit Data Operations	3-29
3.2.6.2	Flight Dynamics	3-31
3.2.6.3	Mission Planning and Scheduling	3-32
3.2.7	Data Processing Facilities	3-32
3.2.7.1	Telemetry On-Line Processing System	3-32
3.2.7.2	Image Processing Facility	3-33
3.2.7.3	Spacelab Data Processing Facility	3-34
3.2.8	Data Accountability System	3-34
3.2.9	Mission Operations Center	3-35
3.2.10	External Experimenters	3-35
3.2.11	Miscellaneous	3-36
3.3	Model POCC Requirements	3-36
3.3.1	NASA/GSFC Mission Model	3-37
3.3.2	MMS Model POCC	3-39
3.3.2.1	Introduction	3-39
3.3.2.2	Model POCC Functional Requirements	3-39
3.3.2.3	Model POCC System Control	3-46
3.3.2.4	Model POCC System Improvements	3-49
3.4	Section 3 References	3-51

SECTION 4

SYSTEMS' ENGINEERING

4.1	General	4-1
4.2	Systems Engineering Analyses	4-1

CONTENTS (Cont)

<u>Paragraph</u>		<u>Page</u>
4.2.1	Systems Drivers	4-2
4.2.1.1	Driver 1: Standard Spacecraft, Shuttle, and TDRSS Orientation	4-3
4.2.1.2	Driver 2: Flexibility	4-3
4.2.1.3	Driver 3: Low Cost of Each POCC	4-5
4.2.1.4	Driver 4: POCC Autonomy	4-5
4.2.1.5	Driver 5: Orientation to People	4-5
4.2.1.6	Driver 6: Advances in Computer Technology	4-6
4.2.1.7	Driver 7: Evolutionary Organization and Operation	4-9
4.2.2	Systems Concept	4-12
4.2.2.1	Analysis of External Requirements	4-12
4.2.2.2	Analysis of Systems Drivers	4-14
4.2.2.3	Synthesis of Systems Concept	4-15
4.2.2.4	Summary	4-20
4.2.3	System Requirements Analysis	4-20
4.3	Systems Studies	4-22
4.3.1	IPC Simulation	4-22
4.3.1.1	Summary of IPC Simulation Results	4-22
4.3.1.2	Further IPC Study Activity	4-23
4.3.2	Systems Implementation Language Study	4-23
4.3.2.1	Summary of Consultant's Recommendations	4-25
4.3.2.2	Study Manager's Recommendations	4-25
4.4	Software Engineering	4-27
4.4.1	Software Engineering Standards	4-28
4.4.2	Common Software Steering Group	4-34
4.4.2.1	Software Engineering Standards Subgroup	4-35
4.4.2.2	Mathematical Analysis and Software Subgroup	4-36
4.4.2.3	Ground Systems Software Subgroup	4-36

CONTENTS (Cont)

<u>Paragraph</u>		<u>Page</u>
SECTION 5		
HOST COMPUTER SYSTEMS		
5.1	General	5-1
5.2	Host Computer Capabilities	5-2
5.3	Applications Processors	5-5
5.4	Data Base Storage System	5-8
5.4.1	Introduction	5-8
5.4.2	Objectives	5-10
5.4.3	DBS Functional Overview	5-11
5.4.4	DBS Hardware Configuration	5-17
5.4.5	Processor Coordination	5-18
5.4.6	Backup and Recovery	5-21
5.4.7	DBS System Software	5-22
5.4.7.1	Vendor-Supplied Operating System	5-22
5.4.7.2	File Management System	5-23
5.4.7.3	Data Base Management System	5-23
5.4.7.4	Pocnet-Developed Systems Software	5-23
5.4.7.5	Software Tools	5-24
5.5	Virtual Interface Processors	5-25
5.5.1	Requirements	5-25
5.5.1.1	Need for Virtualization of I/O Devices	5-25
5.5.1.2	Number of Virtually Interfaced Devices Required	5-25
5.5.2	Functional Definition of Virtual Terminals	5-26
5.5.2.1	Random Access Block Display	5-26
5.5.2.2	Size System	5-27
5.5.2.3	Notational Convention	5-28
5.5.2.4	Text and Commands	5-28
5.5.2.5	Complete Commands	5-28
5.5.2.6	Positioning Controls	5-29

CONTENTS (Cont)

<u>Paragraph</u>		<u>Page</u>
5.5.2.7	Elements-Common to All VT Commands	5-30
5.5.2.8	MODESET Commands	5-30
5.5.2.9	SENSE Commands	5-31
5.5.2.10	PUT Commands	5-32
5.5.2.11	GET Commands	5-33
5.5.2.12	Object Coding of VIP Commands	5-33
5.5.3	Virtual Users -- Functional Definition	5-33
5.5.4	Host Requirements for VIP and Gateway Proc- essors	5-35
5.5.4.1	Multiprogramming and Reentry	5-35
5.5.4.2	String Handling	5-36
5.5.4.3	Control Blocks, Buffers, and Working Storage	5-36
5.5.4.4	Microprogramming	5-36
5.5.4.5	I/O Channels	5-36
5.5.4.6	Timing	5-36
5.6	Gateways	5-36
5.6.1	General Characteristics	5-36
5.6.2	Functional Description of Gateways	5-37
5.6.3	Interfaces To Be Provided Initially	5-38
5.6.3.1	Telops	5-38
5.6.3.2	Johnson Space Center	5-38
5.6.3.3	System 360 at Goddard Space Flight Center . . .	5-39

SECTION 6

INTER-PROCESS COMMUNICATIONS

6.1	Requirements Analysis	6-1
6.1.1	Functional, Operational, and Performance Requirements	6-1
6.2	Design Alternatives	6-2
6.2.1	Common Access Memory	6-2
6.2.2	Crossbar Switching	6-3

CONTENTS (Cont)

<u>Paragraph</u>		<u>Page</u>
6.2.3	Common Bus	6-4
6.2.4	Cable Bus	6-5
6.2.5	Basic Loop Design	6-5
6.2.6	Computer Subnet	6-6
6.2.7	Conclusions	6-7
6.3	Operational Description	6-9
6.3.1	Overview	6-9
6.3.2	General Description of PMPNET Access Protocol . .	6-11
6.4	Pocnet Message Processor Description	6-11
6.4.1	PMP Characteristics	6-11
6.4.2	PMP Software Organization	6-12
6.4.3	Frame Queue	6-12
6.4.4	Message Routing Table	6-14
6.4.5	PMP Statistics	6-17
6.4.5.1	Traffic by Physical Channel	6-17
6.4.5.2	Traffic by LCI	6-18
6.4.6	PMP Timing Estimate	6-20
6.4.7	PMP Core Estimate	6-22
6.5	IPC DOCC Description	6-22
6.6	Establishment of a Minimum Pocnet System	6-23
6.7	User-Level Commands	6-24

SECTION 7

APPLICATIONS ENGINEERING

7.1	General	7-1
7.2	POCC Applications	7-1
7.3	Data Base Storage Applications	7-2
7.3.1	Data, Catalog, and Library Management and Main- tenance	7-3
7.3.2	Software Engineering Management and Configuration Management Tools	7-3
7.3.3	Mission Planning Tools	7-3
7.3.4	Document Preparation and Maintenance Tools . . .	7-8
7.3.5	Computer Conference (Mailbox Service)	7-9

CONTENTS (Cont)

<u>Paragraph</u>		<u>Page</u>
7.4	Simulation	7-9
7.4.1	Requirements	7-10
7.4.1.1	Training	7-10
7.4.1.2	Operations	7-10
7.4.1.3	Analysis	7-11
7.4.2	Design Goals	7-11
7.4.2.1	Payload Simulation/POCC Interfaces	7-11
7.4.2.2	Simulator Installation	7-13
7.4.2.3	On-Line Operations	7-13
7.4.2.4	Off-Line Operations	7-13
7.4.2.5	Resource Sharing and Modularity	7-14
7.4.3	Functional Design of Payload Simulators	7-14
7.4.4	Operational Capabilities	7-18
7.4.4.1	Real-Time and Non-Real-Time Operation	7-18
7.4.4.2	On-Line and Off-Line Configurations	7-19
7.4.4.3	Phases and Modes	7-19
7.4.4.4	Checkpointing and Refreshing	7-20
7.4.4.5	Interactive Control via Simulation Director Console	7-21
7.4.4.6	System Timing and Diagnostics	7-21
7.5	On-Board Computer Support	7-22
7.5.1	On-Line OBC Support	7-22
7.5.2	Off-Line OBC Support	7-23
7.6	Command Memory Management (CMM)	7-24
7.6.1	Interface Information	7-24
7.6.2	Memory Load Structure Information	7-25
7.6.3	Command Loading Information	7-25
7.6.4	Clock Information	7-25
7.6.5	Operational Modes Information	7-26

CONTENTS (Cont)

<u>Paragraph</u>		<u>Page</u>
SECTION 8		
OPERATIONS		
8.1	General	8-1
8.2	Maintenance and Operations	8-1
8.2.1	Maintaining Resources Tables	8-1
8.2.2	Scheduling Resources	8-2
8.2.3	Establishing Configurations	8-2
8.2.4	Verifying Configurations	8-4
8.2.5	Maintaining Status	8-6
8.2.6	Providing Fault Isolation, Repair, <u>and</u> Evaluation . .	8-6
8.2.7	Providing Performance Monitor and System Analysis	8-7
8.2.8	Performing Simulations	8-9
8.2.9	Pocnet Maintenance	8-9
8.3	Data Operations Control	8-11
8.3.1	DOCC Operating Functions	8-11
8.3.2	DOCC Software Requirements	8-12
8.3.2.1	DOCC Operating System	8-12
8.3.2.2	DOCC Applications	8-14
8.3.3	DOCC Hardware Requirements	8-15
8.3.3.1	Interactive Terminals	8-15
8.3.3.2	Visual Aids	8-15
8.3.3.3	Hard-Copy Devices	8-15
8.3.3.4	Communications	8-16
8.3.3.5	Portable Test Stations	8-16
8.3.4	POCC Data Operations Control	8-16
8.4	Facilities	8-17
8.4.1	Pocnet Unique Systems	8-17
8.4.2	Space Requirements	8-19
8.4.2.1	Equipment	8-19

CONTENTS (Cont)

<u>Paragraph</u>		<u>Page</u>
8.4.2.2	Operations	8-19
8.4.2.3	Support	8-19
8.4.3	Functional Layout	8-19
8.4.4	Space Availability	8-21

SECTION 9

USING POCCNET

9.1	General	9-1
9.2	Information	9-1
9.2.1	Pocchnet User Handbook	9-1
9.2.2	Pocchnet User Interface Engineer	9-2
9.3	Formal Requirements	9-2
9.3.1	Configuration Control Board	9-2
9.3.2	Interface Control Documents	9-3
9.4	User Operations Interface	9-3

LIST OF ABBREVIATIONS AND ACRONYMS

APPENDIXES —

APPENDIX 3A

GSFC SYSTEMS TEST AND OPERATION LANGUAGE (STOL) FUNCTIONAL REQUIREMENTS AND LANGUAGE DESCRIPTION

APPENDIX 3B

PAYLOAD DATA BASE CONCEPT

APPENDIX 4A

VIRTUAL TERMINALS FOR SPACECRAFT PAYLOAD OPERATIONS

APPENDIX 4B

POCCNET SIMULATION STUDY

APPENDIX 4C

A STUDY OF SYSTEMS IMPLEMENTATION LANGUAGES FOR THE POCCNET SYSTEM

APPENDIX 4D

POCCNET SOFTWARE ENGINEERING STANDARD APPROACH RECOMMENDATION

APPENDIXES (Cont)

APPENDIX 4E
EVOLUTIONARY DISTRIBUTED SYSTEMS DESIGN

APPENDIX 5A
PAYLOAD OPERATIONS CONTROL CENTER NETWORK
(POCCNET) PROTOCOLS MANUAL

APPENDIX 5B
DATA BASE STORAGE SYSTEM

APPENDIX 5C
CONTROL OF VIRTUALLY INTERFACED PERIPHERALS

APPENDIX 5D
A LAYERED INTERFACE STRUCTURE FOR DISTRIBUTED
SYSTEMS BASED ON X.25

APPENDIX 6A
INTER-PROCESS COMMUNICATION (IPC)
REQUIREMENTS ANALYSIS

APPENDIX 6B
POCCNET "CROSS-BAR" SWITCHING STUDY

APPENDIX 6C
A SAMPLE POCCNET CHANNEL IMPLEMENTATION

APPENDIX 6D
MICROPROCESSOR IMPLEMENTATION OF CCITT RECOMMENDATION
X.25 (LEVELS 1 & 2) PROTOCOL

APPENDIX 6E
IPC DOCC LANGUAGE EXAMPLES

APPENDIX 6F
ESTABLISHMENT OF A MINIMUM POCCNET SYSTEM

APPENDIX 6G
USER LEVEL COMMANDS

APPENDIX 6H
HIGH PERFORMANCE LOCAL COMMUNICATIONS BASED ON THE
CCITT X.25 PROTOCOL

APPENDIX 7A
PAYLOAD SIMULATIONS

APPENDIX 7B
FUNCTIONAL DESIGN OF PAYLOAD SIMULATORS

APPENDIXES (Cont)

APPENDIX 7C FUNCTIONAL DESCRIPTION OF A TYPICAL CMM SYSTEM

APPENDIX 8A STATUS MONITORING SYSTEM

ILLUSTRATIONS

<u>Figure</u>		<u>Page</u>
3-1	MMS-Type Spacecraft Command Flow	3-5
3-2	Scheduling and Status Block Diagram	3-13
3-3	Desired Levels of Support System Test/Verification Capabilities	3-16
3-4	POCC-to-KSC Interfaces (Composite)	3-19
3-5	POCC-to-KSC Interfaces (Payload at PPF)	3-20
3-6	POCC-to-KSC Interfaces (Payload at O&C)	3-21
3-7	POCC-to-KSC Interfaces (Payload at OPF)	3-22
3-8	POCC-to-KSC Interfaces (Payload at the Pad)	3-23
3-9	Data Links, IUS/Payload Within Orbiter Sphere of Influence	3-26
3-10	Data Links, Before IUS/Payload Detachment (TDRS Not Available)	3-27
3-11	Data Links After Synchronous Orbit Achieved	3-28
3-12	Launch Support Period Chart	3-38
4-1	Systems Engineering Approach Used in Poccnet Study . .	4-2
4-2	Endless POCC Systems Evolution	4-12
4-3	Systems Concept Synthesis	4-13
4-4	Summary of the Application of Systems Drivers to the Synthesis of a Systems Concept	4-16
4-5	Autonomous Model POCC	4-20
4-6	Fully Interconnected Model POCC	4-20
4-7	Computerworld Article on Government Acceptance of DOD-1	4-26
4-8	Recommended Composition of Software Engineering Panel	4-30
4-9	Standard Approach Overview	4-31
5-1	Establishing Logical Connections in Poccnet	5-4
5-2	Applications Processors in Poccnet Environment	5-6

ILLUSTRATIONS (Cont)

<u>Figure</u>		<u>Page</u>
5-3	Centralized and Decentralized Data Base Philosophies	5-12
5-4	Poccnnet Data Base Philosophy	5-14
5-5	Proposed Structure of DBS	5-15
5-6	Poccnnet DBS Subsystem Hardware Configuration	5-16
5-7	Overview of Scheduling	5-20
5-8	Virtual User Scenario	5-34
5-9	Gateway Data Paths	5-38
6-1	Common Access Memory	6-3
6-2	Crossbar Design	6-3
6-3	Common Bus Design	6-4
6-4	Loop Design	6-6
6-5	Computer Subnet	6-7
6-6	AP and IPC Subsystem	6-9
6-7	Process-to-Process Communication Sequence	6-10
6-8	End-to-End/Point-to-Point Transmission	6-10
6-9	Point-to-Point Acknowledgement	6-11
6-10	PMP Software Organization	6-13
6-11	Frame Queue in the PMP	6-14
6-12	Source, Network, and Destination LCIs	6-15
6-13	Example of Message Routing Tables	6-16
6-14	Sample Traffic-by-Physical-Channel Table	6-17
6-15	Sample Traffic-by-LCI Table	6-19
6-16	IPC/HOST DOCC	6-23
7-1	POCC Mission Planning Interfaces	7-5
7-2	Payload Simulation/POCC Interfaces	7-12
7-3	Payload Simulations System Structure	7-15
8-1	Formation of POCCs from Poccnnet Resources	8-3
8-2	Connection Verification Over Virtual Channels	8-5
8-3	Desired Levels of Support-System Test/Verification Capabilities	8-8

ILLUSTRATIONS (Cont)

<u>Figure</u>		<u>Page</u>
8-4	Pocnet Conceptual System (Model POCC Pair, DOC/ DBS Complex, and Interfaces)	8-18
8-5	Pocnet Functional Layout	8-20

TABLES

<u>Table</u>		<u>Page</u>
5-1	Levels of Service	5-18
6-1	Requirements and Design Alternatives	6-8
6-2	CPU Degradation	6-20
7-1	Summary of Simulation Types Versus Simulation Com- ponents and Characteristics	7-17

SECTION 1
INTRODUCTION

SECTION 1

INTRODUCTION

Changing conditions both within NASA and in the external environment require more uniform system concepts for the GSFC POCCs.

Within NASA, the new standard payloads (such as MMS), standard launch system (Shuttle), and standard communications systems (TDRSS) will inevitably lead to an increasing number of missions to be supported in the 1980s. Most of these missions will have lower budgets and shorter lead time for POCC development than typical missions of the past. Consequently, future POCC subsystems must possess a high degree of flexibility so that they may be reused to save development time and money. In addition, the POCCs must be organized to allow evolutionary changes, because requirements generated by other mission elements will certainly change during the coming decade.

In the external environment, conditions that affect POCC development are also changing. Principal among these are increasing costs for development and operations personnel, continually decreasing hardware costs, standardization of communications protocols, and availability of sophisticated software for low-cost minicomputers. In response to these trends, the POCCs must be oriented toward solving human problems in order to decrease program development, training, and operations expenses. Additionally, the POCCs should be implemented as a collection of standard hardware systems, each specialized to handle a given application, connected by high-speed serial channels, and using standard software systems.

Notwithstanding the rapidly changing environment of mission support systems, one POCC requirement remains constant: to provide for absolute payload safety. To meet this prime requirement, GSFC must continue to provide autonomous POCCs. The Poccnnet approach to satisfying this goal in light of these changing conditions is described in this report and its appendixes.

Section 2 of this report is an executive summary of sections 3 through 9. It concisely presents the issues raised and the conclusions reached in each of those sections. Section 3 describes the interface and support requirements placed on the GSFC POCCs and outlines the Model POCC concept. In Section 4, the operational requirements presented in Section 3 together with generic requirements are analyzed to arrive at a system concept for GSFC payload operations. This concept, as was

previously mentioned, involves the use of a number of standard minicomputers to perform the various tasks necessary for POCC operations and support. These mini-computer systems, called host computer systems, are described in Section 5. The system concept calls for the interconnection of the host computer systems by a number of high-speed serial communications channels, which make up the Inter-Process Communication (IPC) system, which is described in Section 6. The name Pocnet is derived from the resulting network of POCCs. Section 7 proposes a number of applications to be developed for the various host computer systems, including common software modules, data base applications, simulations, command memory management, and on-board computer support. Section 8 presents the operational and facilities requirements for Pocnet. Section 9 discusses the use of Pocnet and the definition and control of interfaces.

The appendixes referenced throughout this report consist of various documents and other reference material which give more detailed information on the study areas. These appendixes are published under a separate cover from the main body of this report.

SECTION 2
EXECUTIVE SUMMARY

SECTION 2

EXECUTIVE SUMMARY

CONTENTS

<u>Paragraph</u>		<u>Page</u>
2.1	General	2-1
2.2	Systems Requirements	2-1
2.2.1	Shuttle Mission Control Center	2-1
2.2.2	NASCOM and STDN	2-2
2.2.3	Kennedy Space Center	2-4
2.2.4	IUS/SSUS Operations Control Center(s)	2-4
2.2.5	Spacelab Payload Operations Control Center	2-5
2.2.6	Operational Support Computing	2-5
2.2.6.1	Orbit Data Operations	2-5
2.2.6.2	Flight Dynamics Computing	2-6
2.2.6.3	Mission Planning and Scheduling	2-6
2.2.7	Data Processing Facilities	2-7
2.2.7.1	Telemetry On-Line Processing System	2-7
2.2.7.2	Image Processing Facility	2-7
2.2.7.3	Spacelab Non-Time-Critical Data Processing	2-7
2.2.8	Data Accountability System	2-8
2.2.9	Mission Operations Center Mission Control Room	2-8
2.2.10	External Experimenters	2-8
2.2.11	Miscellaneous Interfaces	2-9
2.2.12	Model POCC Requirements	2-9
2.3	Systems Engineering	2-10
2.3.1	Systems Engineering Approach	2-10
2.3.1.1	Generic System Drivers	2-10
2.3.1.2	Synthesis of Systems Concept	2-14
2.3.1.3	Quantitative Requirements	2-16
2.3.2	Systems Studies	2-17
2.3.3	Software Engineering Standard Approach	2-18
2.4	Host Computer Systems	2-18
2.4.1	Applications Processors	2-19
2.4.2	Data Base Storage System	2-20

CONTENTS (Cont)

<u>Paragraph</u>		<u>Page</u>
2.4.3	Virtual Interface Processor	2-21
2.4.4	Gateways	2-22
2.5	Inter-Process Communication System	2-23
2.5.1	Inter-Process Communication Requirements Analysis	2-23
2.5.2	Design Alternatives	2-24
2.5.3	IPC System Description	2-24
2.6	Applications Engineering	2-26
2.6.1	DBS Applications	2-26
2.6.2	Simulation	2-27
2.6.3	On-Board Computer Support	2-27
2.6.4	Command Memory Management	2-28
2.7	Operations	2-29
2.8	Using Poccnet	2-29
2.8.1	Obtaining Information About Poccnet	2-30
2.8.2	Formal Requirements	2-30
2.8.3	User Operations	2-30

SECTION 2

EXECUTIVE SUMMARY

2.1 GENERAL

This section provides a summary of the remainder of the study report. An overview of Sections 3 through 9 is presented with references to applicable appendixes or other sections of the report.

2.2 SYSTEMS REQUIREMENTS

Section 3 describes the requirements analysis that was carried out under the Pocnet Definition Phase Study. The analysis identified requirements for the following 11 interface areas for the POCCs in the 1980s:

- a. Shuttle Mission Control Center (SMCC).
- b. NASCOM and STDN.
- c. Kennedy Space Center.
- d. IUS/SUSS Operations Control Center.
- e. Spacelab Payload Operations Control Center.
- f. Operational Support Computing.
- g. Data processing facilities.
- h. Data Accountability System (DAS).
- i. Mission Operations Center Mission Control Room.
- j. External experimenters.
- k. Miscellaneous interfaces.

Each of these areas is discussed briefly in paragraphs 2.2.1 through 2.2.11 and the Model POCC requirements are discussed in paragraph 2.2.12.

2.2.1 SHUTTLE MISSION CONTROL CENTER

The Shuttle Mission Control Center (SMCC), located at Houston, Texas, is responsible for STS flight operations and control.

The SMCC/POCC interface data functions are as follows:

- a. From SMCC to POCC
 - (1) Payload data in OI downlink; a minimum of 1 kbps cleaned-up house-keeping telemetry and 2 kbps tape recorder playback per payload (five payloads per flight).
 - (2) Time correlated state vectors on Orbiter orbit and attitude.
 - (3) Time correlations between Orbiter clock, spacecraft clock, and ground time.
 - (4) Command history; command history buffer (up to 200 commands) in real-time and full history after touchdown.
 - (5) Data base information.
- b. From POCC to SMCC
 - (1) Commands (verification loop back).
 - (2) Data base information including planning.

All data received from, and sent to, SMCC will be in 4800-bit NASCOM blocks. No block will contain data for more than one payload. In the case of multispacecraft, SMCC will demultiplex payload telemetry into separate NASCOM blocks; the POCC will be responsible for queuing commands and distributing payload telemetry and operational data.

2.2.2 NASCOM AND STDN

The NASA Communications Network (NASCOM) and the Spacecraft Tracking and Data Network (STDN) are global systems established and operated by NASA to provide operational communications and tracking support, respectively, of all NASA space payloads.

By the end of 1979, the Tracking and Data Relay Satellite System (TDRSS) of STDN will become fully operational. The TDRSS era will be characterized by the bent-pipe throughput mode of telecommunications. Although it is planned that all ground STDN (GSTDN) stations will ultimately operate in a TDRSS-compatible bent-pipe mode, it is anticipated that they will provide their present service as long as it is required and it remains feasible to do so.

TDRSS will provide low-rate downlink service of rates not exceeding 50 kbps for 20 payloads simultaneously and uplink service of rates not exceeding 10 kbps for 2 payloads simultaneously, with two data channels per return link. For higher data rates there will be two forward and two return links which will have rates of up to 300 kbps and 6 Mbps, respectively, and two forward and two return links which will have rates of up to 25 Mbps and 200 Mbps, respectively, with four data channels per return link.

The baseline telemetry service bandwidth will be 1.544 Mbps, partitioned into 56-kbps segments. Any multiplexing/demultiplexing will be transparent to the POCC. TDRSS will supply blocked and convolutionally decoded telemetry data in a throughput mode, but NASCOM will provide a 1.544-Mbps 30-minute contingency recording capability. There will be no frame sync detection of payload data, and time-tagging will be by the time of receipt at the TDRSS contractor/NASCOM interface of the first bit of data in a NASCOM block.

The POCC will input its blocked and formatted command data to NASCOM, which will route the commands to the GSTDN or TDRSS where they will be immediately throughput to the spacecraft. There will be a command echo block returned to the POCC by NASCOM.

The Network Control Center (NCC) will provide the following real-time support interface between the POCCs and the STDN:

- a. Real-time and emergency scheduling.
- b. Data monitoring and service accountability.
- c. Testing and simulations involving STDN resources.
- d. Fault isolation.
- e. Ground configuration commands.

The NCC/POCC interface data functions are as follows:

- a. From POCC to NCC
 - (1) Schedule requests (generic or specific).
 - (2) Ground system directives.
 - (3) Support quality evaluation report.

- (4) Data base access for resource characteristics, availability, etc.
 - (5) Simulation data (telemetry and command).
- b. From NCC to POCC
 - (1) Schedule of assigned resources.
 - (2) Performance data (particularly for TDRSS).
 - (3) Data base access for resource characteristics, availability, etc.
 - (4) Simulation data (telemetry and command).

2.2.3 KENNEDY SPACE CENTER

Launch operations at the Kennedy Space Center (KSC) for expendable launch vehicles are on-going and well defined. This description defines the KSC/Shuttle/POCC interfaces as they are believed to be at the present time.

The KSC/Shuttle/POCC interface data functions are limited to final integration and checkout of telemetry and command data transfers. The POCC will be interfacing with KSC by means of the NASCOM/POCC and SMCC/POCC interfaces described in paragraphs 2.2.1 and 2.2.2. Therefore, the POCC expects to transmit and receive only blocked data. Note also that after payload checkout and flight preparations are completed (that is, after the payload leaves the Payload Processing Facility), all POCC/KSC data communications will be via SMCC.

2.2.4 IUS/SSUS OPERATIONS CONTROL CENTER(S)

The Interim Upper Stage (IUS) spacecraft, which is three-axis stabilized and has on-board computers, will be capable of delivering a 1500- to 5000-pound Orbiter-launched payload to geosynchronous orbit as well as to other planets. The IUS will be tracked by GSTDN and, at separation, will supply separation pyrotechnic firing power.

The IUSOCC/POCC interface data functions are as follows:

- a. From IUSOCC to POCC — IUS motor ignition times and IUS orbit and attitude ephemeris and trajectory data in near-real-time.
- b. From POCC to IUSOCC — Orbit and attitude of the IUS/payload prior to first IUS motor burn and those orbital parameters which define the desired geosynchronous orbit.

The Spinning Solid Upper Stage (SSUS) spacecraft will be pointed by the Orbiter and subsequent SSUS action will be autonomous. NASA is responsible for tracking the SSUS and providing attitude and orbit determination. A payload coupled with an SSUS must have an apogee kick motor.

2.2.5 SPACELAB PAYLOAD OPERATIONS CONTROL CENTER

The GSFC POCC interfaces required to support experimentation on Spacelab are highly dependent on the capabilities of the Spacelab POCC facility at JSC. It is likely that several ASP, AMPS, and/or EVAL experimenters will require the capability to support simulations training and/or operations from GSFC, since this center is providing experiment integration facilities for these payloads. Such a capability is called a Remote Information Center (RIC). In support of this capability, an interface will be required between the Spacelab POCC and the GSFC experimenters for transmitting relevant Orbiter and Spacelab data, orbit and attitude information, and instrument housekeeping and quick-look analysis outputs. In addition, command capability in real-time will be required between experimenter and the Spacelab POCC. The number of experimenters or instruments requiring this support mode could range between 0 and 10 for any ASP, AMPS, and/or EVAL mixed or dedicated mission.

2.2.6 OPERATIONAL SUPPORT COMPUTING

2.2.6.1 Orbit Data Operations. The POCC/orbit data operations data functions as follows:

- a. From POCC to orbit data operations.
 - (1) Stripped telemetry for navigational analysis and on-board computer (OBC) orbit load.
 - (2) POCC schedules.
- b. From orbit data operations to POCC.
 - (1) Orbit ephemeris.
 - (2) OBC orbit loads.
 - (3) Schedule requests for metric data.

2.2.6.2 Flight Dynamics Computing. The POCC/flight dynamics computing data functions are as follows:

- a. From POCC to flight dynamics computing.
 - (1) Real-time telemetry parameters on a case-by-case basis.
 - (2) Attitude-related data.
 - (3) Mission and/or class-dependent data.
 - (4) Real-time Orbiter (and/or IUS/TUG) attitude and orbit ephemeris.
 - (5) Spacecraft health and safety parameters.
 - (6) Remote operations capability for Poccnet data bases.
 - (7) Report generation, such as maneuver requirements and desired activity timelines.
- b. From flight dynamics computing to POCC.
 - (1) Real-time computed attitude results and information (for example, gyro bias and spin vectors).
 - (2) Payload command information for implementing maneuver plans.
 - (3) Desirability of maneuver summary reports.
 - (4) Flight-dynamics-related POCC OBC support information (for example, star catalog updates).
 - (5) Remote operations capability for Poccnet data bases.
 - (6) Report generation, such as planned payload maneuver and activity sequence.

2.2.6.3 Mission Planning and Scheduling. Mission planning and scheduling aids (such as scheduling matrices and world maps) are now provided to the POCCs on a paper medium as are the scheduling aids generated by Command Memory Management. POCC requirements for mission planning and scheduling are the subject of an ongoing joint study with the Operations Support Computing Division. To meet the needs of the POCC mission planners, the following minimum requirements must be met:

- a. To access mission planning in a mission planning data base.
- b. To access scheduling capability in a STDN scheduling data base.
- c. Possible special scheduling aids generated on request.

2.2 7 DATA PROCESSING FACILITIES

2.2.7.1 Telemetry On-Line Processing System. The Telemetry On-Line Processing System (Telops) is a NASA telemetry data collection and distribution system that is located at GSFC. Telops provides for initial capture of the telemetry data for recording in a form for subsequent use, for decommutation of the processed telemetry into groupings relevant to a particular experiment, and for arranging for the transmittal of these data groups to the experimenters by mailed tapes or possibly by communications lines.

The Telops input processor will be capable of supporting a maximum 2.688-Mbps input rate; however, the total volume of processing that is possible is 4.5×10^9 bits per day of telemetry data, which is an average of 52 kbps. The Telops mass storage system will hold 1.3×10^{12} bits of data (8×10^{11} telemetry bits).

Although presently in the formative stage, Telops envisions having an output processor system which will utilize communications lines for data transfer to the experimenters and will provide a conversational request mode to use when requesting data. If such a system is implemented, a POCC/Telops interactive interface will be easy to effect.

2.2.7.2 Image Processing Facility. The major function of the Information Processing Facility (IPF) will be processing image data. The only interface between IPF and the Telops will be that Telops will deliver tapes of image data to IPF to be processed. No other IPF/Telops interface is planned.

The current interface between IPF and POCC is the requirement from IPF for certain housekeeping data from the LandsatOCC. IPF feels that sensor status and mode data will be required by IPF in the future.

2.2.7.3 Spacelab Non-Time-Critical Data Processing. This processing includes data reduction, formatting, and merging of the data with orbit and attitude information. The latter merging process may also require inputs from the mission command history and support computing systems. Processing of this type will be undertaken at the GSFC facility, and the raw data will be received at GSFC directly from TDRSS. The additional information required for the merging process is available at the Spacelab POCC, although more refined attitude information may be required for some experiments. If a more detailed analysis is required for certain types of information, this would normally be undertaken by IPD at GSFC.

2.2.8 DATA ACCOUNTABILITY SYSTEM

In the 1980s, the Data Accountability System (DAS) will collect information and compile a data base for evaluating the quality and quantity of payload support and to provide management information. Inputs to the DAS from the POCCs and other users may be in real-time and will consist of the following:

- a. Schedule information.
- b. Support analysis data derived from examining the telemetry stream.
- c. Other yet undefined data accountability reports.

Recent studies are focusing on defining more clearly the appropriate role of DAS in the 1980s.

2.2.9 MISSION OPERATIONS CENTER MISSION CONTROL ROOM

The Mission Control Room (MCR) of the Mission Operations Center (MOC) requires mission-related information during launch and special-maneuver phases of missions. The primary method of presenting the information required from a POCC to the MCR will be via the closed-circuit television (CCTV) system. Any normal television video output from the POCC can be patched and shown in the MCR.

2.2.10 EXTERNAL EXPERIMENTERS

Interfaces will be required to support command requests and data handling functions for external experimenters for many payload programs to be supported by the GSFC POCCs. The experimenter facilities used in these support modes may be either based at GSFC or the home institution. In the case of experimenter facilities based at the home institution, interfaces could be required to support either the transmission of real-time command request messages to the POCC or interactive communications between the experimenter and a payload support computing system at GSFC. Types of information possibly required by an experimenter at his facility include raw data dumps, instrument housekeeping and quick-look analysis outputs, relevant payload parameters, and orbit/attitude data.

The development of GSFC-based facilities for quick-look scientific data interpretation and payload control is expected for a number of proposed payload programs. These quick-look facilities would be located separately from the POCC but would be required

to interface with the POCC in order to transfer payload commands or control requests. GSFC-based facilities of this type would require access to the instrument data stream and to relevant payload, orbit, and attitude data generated or processed at the POCC. Communication lines from the quick-look analysis facility could be required to link one or more remote terminals located at the experimenter home institutions directly with the system.

2.2.11 MISCELLANEOUS INTERFACES

The Poccnnet Inter-Process Communication system (refer to Section 6) proposes to use the Department of Defense ARPANET system philosophy together with the new national/international proposed standard serial communication protocols. By so doing, Poccnnet would be able to communicate easily with the world; that is, Poccnnet could communicate readily with any demonstration or operational device which is on a network with standard protocols.

In the realm of public relations, such a Poccnnet could support terminals in the Visitor Center, real-time displays at the Sioux Falls Demonstration Center, special remote terminals during launch, etc.

Such a Poccnnet would easily allow GSFC to support remote integration and test facilities, light-weight portable POCCs, remote POCCs, etc. The possibilities are endless.

2.2.12 MODEL POCC REQUIREMENTS

The Model POCC concept is an essential part of the Poccnnet approach. The Model POCC is an abstraction or pure design of the capabilities all POCCs must have and how they should be implemented in the Poccnnet era. The Model POCC starts as only a paper design representing the best of existing POCC design experience, but it serves as a guideline for POCC developers to show them how common functions are to be implemented in Poccnnet POCCs. Thus, all Poccnnet POCCs will share basically the same design, while still responding fully to the payload project requirements. As Poccnnet Model POCCs are developed at GSFC, the paper design will be transformed gradually to a set of standard software and hardware components and subsystems which will fit easily together because they were system engineered according to a common model. In this way, maximum commonality among POCC subsystems can be achieved without unduly constraining POCC designers. Model

POCC activity in the Poccnet Project is embodied in the Systems Engineering Standards and the Common Software Modules Program.

Functional capabilities required in an MMS Model POCC are presented in paragraph 3.3.2.

2.3 SYSTEMS ENGINEERING

Section 4 of this report describes the results of systems engineering activities undertaken during the study phase. The most important of these results are as follows:

- a. A total systems concept which satisfies the identified payload support requirements.
- b. A decomposition of the system into a number of independent subsystems.

Each of these subsystems must satisfy its own requirements and is itself a "system".

2.3.1 SYSTEMS ENGINEERING APPROACH

The systems engineering approach adopted for this study consists of three steps. First, the external requirements are identified. Next, generic system drivers are considered. These are generalized objectives or constraints which must be satisfied to ensure that POCC systems evolve to meet future requirements. This step narrows the field of candidate systems concepts to those which can meet the requirements and respond to the drivers. Finally, the external requirements are analyzed and applied to each of the systems concept candidates, producing functional and operational requirements on the individual systems of each candidate. This step also results in the identification of tradeoff studies which must be performed to choose the concept to be used.

The external requirements have been presented previously. The generic system drivers, the various candidate systems concepts, and the application of the requirements to the candidate concepts are briefly described in the following paragraphs and are presented in more detail in Section 4.

2.3.1.1 Generic System Drivers. The following seven system drivers are considered essential to the success of the GSFC POCCs in the 1980s:

- a. Standard spacecraft, Shuttle, and TDRSS orientation.
- b. Flexibility.

- c. Low cost.
- d. POCC autonomy.
- e. People orientation.
- f. Advances in computer technology.
- g. Evolutionary organization and operation.

For standard spacecraft, Shuttle, and TDRSS orientation, the POCCs must be oriented to the Multimission Modular Spacecraft (MMS), Applications Explorer Mission (AEM), Space Shuttle, and TDRSS. These are all new NASA systems which will provide more capability at lower cost than their respective predecessors. The POCCs of this era must also provide increased capability at reduced cost by the use of standard hardware, standard software, and standard implementation and operations procedures. If standard POCC systems and applications are properly designed and implemented initially, they can be reused with, at most, minor modifications for support of other missions. Thus, as in the standard flight systems, the cost is amortized over a number of missions instead of being borne fully by each user.

The second driver identified is flexibility. Because computer hardware invariably outlasts its first application in a POCC, hardware systems must be flexible to permit reuse in future POCCs. Likewise, POCC software must have sufficient flexibility not only to be used on several missions but possibly with different hardware. Thus, long-term flexibility is an important driver of future POCC systems and software. Likewise, short-term flexibility is necessary to permit support of special requirements during launch, critical maneuvers, or contingency operations. Flexibility of the entire POCC systems concept is important so that fluctuations in support requirements can be accommodated in a timely and economical manner. This is especially important in meeting the short lead-time requirements of shuttle-launched payloads.

Low cost of each POCC is the third driver identified in the study. POCCs have traditionally cost only a few percent of the total cost of the spacecraft series being controlled. In the 1980s, as the cost of standard flight systems decreases, GSFC will be under severe pressure to decrease the cost of POCCs.

The usual technique used to provide low cost and flexibility for a number of applications is to pool resources and draw on them according to need. Opposed to this technique is the fourth system driver, POCC autonomy. The POCC should never

have to jeopardize its spacecraft because of the constraints of pooled operation. Each project should always be able to operate its POCC as it deems necessary for legitimate spacecraft health and safety reasons without being unduly concerned about the operations taking place in other POCCs. Therefore, POCC autonomy is an essential system driver.

The fifth driver is people orientation. The direct personnel costs of implementing and operating a POCC for an extended period of time far exceed the equipment costs. The maximum impact that can be made on costs is in the personnel area. Attacking personnel costs head-on means that systems must be oriented not only to technical requirements but also to the requirements of the people who have to develop, use, and maintain them. A system has to provide the man-machine interfaces and capabilities each user needs to be productive on the system. Therefore, people orientation is a necessary system driver.

The sixth system driver is advances in computer technology. This driver declares that recent advances in technology must be brought to bear on GSFC POCC systems of the future. Two effects of recent advances in this area are the rapidly decreasing cost of the digital logic elements which make up computers and the rapidly decreasing cost of computer interconnection via high performance bit-serial channels. The end result of falling logic prices is that it has become economically feasible to distribute a number of computers to solve complex distribution problems. By solving each subproblem where and as it arises, the overall system problem never becomes overwhelming or unmanageable. But to make full use of this distributed computing concept, it must be possible to interconnect the distributed computers easily and at low cost. This is where the new high performance bit-serial synchronous communications technology is so important.

GSFC experience with the SCADI system, which hosts a variety of serial channels at rates up to several million bits per second, has shown the desirability of serial channels. Serial channels provide the simplest, most reliable, most economical means of communication and can be used over a wide range of data rates, at short or long distances. Furthermore, international standards in serial data communication were adopted by ISO in 1977.

All the computer vendors will offer this standard interconnection technology as part of their product line, and their operating systems will support distributed computation structures to solve distributed problems as a matter of course. This technology,

together with the falling cost of digital logic, will have an enormous impact on systems engineering. Therefore, Poccnet is driven by advances in computer technology.

The seventh and final system driver is evolutionary organization and operation of POCC systems. GSFC has over 15 years of experience in designing, building, and operating POCCs. Many of the existing GSFC POCCs had their genesis in the early 1960s. In those days, each new series of spacecraft had radically different characteristics and requirements. Also, the space business was in its fledgling years, and there were honest differences in philosophy as to how a spacecraft was best operated. As a result, each new spacecraft series led to a new POCC development. In addition to the POCCs dedicated to each spacecraft series, there evolved one POCC, the Multi-Satellite Operations Control Center (MSOCC), to serve all those spacecraft which did not require a dedicated POCC because of the nature of their support requirements. The GSFC POCCs have been fully utilized over the years, and each POCC has responded efficiently to the requirements of its spacecraft series. Although the existing dedicated POCCs and the MSOCC have been successful, further evolution of POCC organization at GSFC is now required. This is because a number of POCC systems drivers are presently converging on GSFC for which the existing systems organization is unable to respond effectively.

The GSFC POCC system of the future must be evolutionary in the following three distinct, important ways:

- a. The system must represent an evolutionary growth from existing systems. The GSFC approach to POCCs has worked well in the past and, therefore, must not be abruptly abandoned. The strengths of the existing systems (principally POCC autonomy and responsiveness to project requirements) must be retained. As new POCCs are required, however, small changes from existing systems approaches must be made to increase flexibility, decrease cost, and incorporate standard solutions to common requirements.
- b. Any new systems which are required must be designed to allow evolutionary growth. This characteristic of systems may be called "modular responsiveness to unknown requirements". A system designed for evolutionary growth would allow modest new requirements to be met by small increments in system capability, while more difficult new requirements might create larger perturbations. However, if the system is properly

designed to allow evolutionary adaptation, small changes in requirements should never cause major system perturbations. (This driver is the death knell to large, centralized, integrated POCC systems.)

- c. The total GSFC POCC system must allow evolutionary operation. Too often in the past, systems operation has received little or no attention from systems architects. Yet many of the best recent systems have been designed by writing the user manual first, in effect, establishing operational user specifications as the principal system specifications.

Future GSFC POCC systems must extend this trend to the point of including evolutionary systems operation as a system requirement. This means that systems must be designed to be operated in a simple, straightforward way as the basic operational mode. In addition, the system must be designed to allow the operations personnel to improve systems operation by evolving extensions to the basic operational modes. These extensions would come about through operational experience rather than being simply the ideas of systems designers who do not really know the operational problems.

This approach to operations design of systems, people, and machines is called "automatability" by study personnel. As contrasted with "automated", it implies that operations personnel, rather than design personnel, define the segments to be automated and the level of automation. This allows systems operation to evolve out of operational experience, for efficiency and reliability.

The evolutionary development and operation of new systems must come about by endlessly evaluating and modifying existing systems and creating compatible new systems to meet emerging requirements. Therefore, Poccnet is driven by evolutionary organization and operation. This concept is discussed further in Section 4 and Appendix 4E.

By analyzing these seven system drivers along with the external requirements, a system concept was synthesized.

2.3.1.2 Synthesis of Systems Concept. The starting point for a POCC systems concept for GSFC is an evolutionary small step from current GSFC practice; that is, dedicating a standard minicomputer for a period of time (an hour or a year) to run

the POCC applications. The name applications processor (AP) has been adopted for this processor. It hosts all the POCC applications for a single given payload for the period of time it is assigned. The difference between so-called dedicated POCCs (such as IUE) and so-called multisatellite POCCs (such as MSOCC) lies then in the length of time these APs are dedicated to a particular payload at any one time. The drivers of standardization, flexibility, and low cost imply that there should be a number of identical APs at GSFC, each one able to run standard common software customized by a data base for a particular payload.

Given the allocation of a POCC to be supported on a standard minicomputer, the issue of how to deal with interface complexity arises. The clear answer from a manageability point of view (if it is affordable from a hardware and software point of view) is to isolate nonstandard interfaces into "virtual interface" processors, so that each POCC essentially consists of a core of standard applications software running on a standard AP, surrounded by other processors which handle interface idiosyncracies and present a standard interface to the AP.

These interfaces (identified during the study and discussed in Section 3) break down into the following three groups:

- a. NASCOM interfaces, including telemetry, commands, Shuttle MCC, KSC, and the Spacelab POCC.
- b. Mission operations room (MOR) interfaces, such as keyboard/CRT terminals, printers, etc.
- c. External system interfaces, including mission support computers, Telops, external experimenters, NCC, and DAS.

The system concept meets the requirements of these three interface groups with three separate systems, each optimized to perform its particular job efficiently and economically without impacting the others. These systems are the telemetry and command system (TAC) for the NASCOM interfaces, the virtual interface processors (VIPs) for MOR interfaces, and gateways for external interfaces.

The TAC will handle payload telemetry and commands on a STDN bent pipe. In addition, it will provide interfaces between the POCC and SMCC, KSC, or Spacelab POCC during Shuttle or Spacelab preflight or flight operations. A brief description of the TAC is given in Section 5.

Each POCC MOR contains a variety of computer peripherals which are used to provide the human-to-machine interfaces for payload operations and system control. The software required to control these devices differs among the various device types and among devices of the same type but different manufacturer. A VIP isolates the details of the particular devices which an MOR contains. The VIP does this by mapping messages from the POCC AP which are in a standard generalized format into messages which are coded for a specific device and vice-versa. In this way, the standard applications software on an AP can always drive standard virtual terminal formats in the MOR, no matter what manufacturer's devices are in use there.

The third group of POCC interfaces identified is the external interfaces, such as those to mission support computers, Telops, DAS, and NCC. Each of these systems has unique features which may change from time to time and which are isolated from the POCCs by gateway processors.

To take full advantage of these standard interfaces, it is necessary to have a fast and simple means to form connections among the various POCC APs, gateways, TACs, and VIPs. The Poccnet system that provides this service is the Inter-Process Communication (IPC) system. IPC is described in detail in Section 6 of this report.

Standard POCC software and systems will be adapted to the needs of a specific payload by use of a customizing data base. During POCC operations, this data base will reside on the AP. However, to provide the capability to back up the POCC on another AP and to provide access to the data base for maintenance and non-real-time operations, the data base must also be available on a system other than the AP. To satisfy these requirements, a designated data base storage (DBS) system will be available to provide continuous accessibility to on-line storage for several POCC data bases. The DBS will also host planning and software engineering tools required for POCC development.

2.3.1.3 Quantitative Requirements. The functional requirements identified in Section 3, together with the generic system drivers identified in paragraph 2.3.1.1, are responsible for the decomposition of the total system into the component systems described in paragraph 2.3.1.2. The quantitative requirements identified during the study are: 12 to 30 MORs at GSFC in the 1980s; 4 to 20 terminal devices per MOR; a maximum real-time housekeeping rate of 64 kbps; maximum science rate of

512 kbps, with a few exceptions. The further development of functional and performance requirements is documented in Sections 5 and 6.

2.3.2 SYSTEMS STUDIES

Most of the Poccnet system concept is well within the current state of the art of systems engineering, but two areas requiring specific study were identified. These are the IPC design and the selection of a programming language to be used to implement systems software.

To validate the IPC design, a simulation model of the proposed system was developed. This model was then used to simulate Poccnet operations under a variety of data handling loads on several possible configurations of the IPC subnet. This study did in fact verify that the IPC system could easily support realistic POCC data traffic. In fact, the performance of the subnet nodes was so impressive (although not unexpected) that, following the study, the systems engineering decision was made to integrate the communications subnet node for each MOR into the VIP. (The VIP was similarly found to be woefully underutilized in the original concept because of the high level of autonomy of the low-cost intelligent terminals widely available in 1977.) These results are summarized in paragraph 4.3.1 and are described in greater detail in Appendix 4B. Paragraph 4.3.1 also describes current studies of alternative techniques for implementation of IPC.

In the study of systems implementation languages, directed by Dr. Victor Basili of the University of Maryland, 15 candidate languages were evaluated for suitability in implementation of Poccnet systems. The recommendations of this study are presented in paragraph 4.3.2, along with recommendations of the Poccnet Study Manager based on recent developments in this field. Briefly, the conclusion reached is that GSFC should take advantage of the common High Order Language (HOL) activity of the Department of Defense (DOD). The DOD common HOL will be very similar to several languages which were highly recommended by the University of Maryland study but will have the additional advantages of standardization, widespread availability, and continuing well-funded support from DOD. Until the common HOL is available, Fortran, in conjunction with a suitable structured programming preprocessor, should be used for Poccnet software development.

2.3.3 SOFTWARE ENGINEERING STANDARD APPROACH

Software engineering is a relatively new discipline concerned with the development of high-quality software by the skillful application of software development tools and methods based on a sound understanding of basic principles. To lay the groundwork for the development of cost-effective, reliable, reusable software for GSFC control centers, a study of software engineering methodologies was made as part of the Poccnnet definition phase study. This effort resulted in the development of a standard approach for software engineering which is described in paragraph 4.4 and in Appendix 4D.

The approach developed is a flexible one, which can be tailored to each individual software project's needs. The basis of the standard approach is a six-phase system life cycle consisting of system requirements and specification, functional design, detail design, implementation, system testing and acceptance, and operations and maintenance. This breakdown provides increased visibility of the development effort to management and helps maintain close control.

The standard approach also recommends techniques to be used during the various phases of the system life cycle. These include top-down development, modularity, design and code inspections, and staged implementation.

The Common Software Steering Group (CSSG) is currently evaluating the standard approach developed under this study as well as other software engineering methodologies. The CSSG is a committee composed of representatives from the MMS project and operational support systems elements within the Mission and Data Operations Directorate. In addition, the CSSG is working to establish a mechanism for gathering, storing, and retrieving high-quality, thoroughly documented software products.

2.4 HOST COMPUTER SYSTEMS

Host computers on Poccnnet provide the distributed processing nodes which support individual POCCs and provide specialized nodes to promote resource sharing. Host computers range in size from a dual-processor DBS system providing complex data handling capabilities to relatively small single-purpose minicomputers and perhaps even microprocessors. Most of the host computers on Poccnnet will be components of Model POCCs. Each Model POCC will consist of one AP, one or more

VIPs, and a TAC. Arranged in clusters to provide backup capabilities, these Model POCCs will handle the workload of the GSFC control centers in the 1980s.

Each host computer will be connected to the other computing nodes in Poccnet and to the outside world through the Poccnet IPC system. The procedures to be followed in order to connect a host computer to Poccnet would be described in a Poccnet Protocol Manual. A draft Prototype Poccnet Protocol Manual was developed during the study and is attached as Appendix 5A. Discussions of the following topics are included in this manual:

- Serial line electrical characteristics.
- Point-to-point physical communication.
- Flow control in logical connections.
- End-to-end communication protocols.
- The host interface.
- The user process interface.
- Process-to-process protocols.

2.4.1 APPLICATIONS PROCESSORS

Applications processors (AP) on Poccnet are general-purpose computing systems (computer plus operating system) capable of providing a process-oriented computing environment for applications, together with access to the Poccnet IPC system. In addition, each AP supports the minimum level of Poccnet interoperability conventions, by which the hosts on Poccnet can be operated by a remote user.

APs are provided by individual systems within Poccnet. For example, in one proposed initial Poccnet system approach, MSOCC provides four APs (the XDS 930 replacements), HEAOCC provides two APs (also XDS 930 replacements), the DBS system provides two APs, and the DOCC provides one AP. In addition, the Mission Support Computing and Analysis Division PDP-11/70 real-time simulation system and the 360-65 would be interfaced as APs. The 360-65 would support simulation activities, command memory management, and on-board computers (OBCs). The APs together with the other Poccnet host computer systems and IPC make up a distributed computing system which will meet the requirements for low-cost, re-usable, flexible ground support systems at GSFC in the 1980s. Implementation approaches for Poccnet are discussed in a separate program plan.

Each AP has the capacity of, for example, a PDP-11/70 and is capable of running all the applications programs in a modern POCC (like AEOCC). APs are not special-purpose hardware but are general-purpose computers bought from a computer manufacturer together with a multiprogramming real-time operating system. Depending on how they are configured, APs could cost from \$100K to \$400K in 1975, and from \$10K to \$100K in 1985. They are the computing nodes which "host" general computing pieces of a distributed computing system, and they are the applications processing subsystems of Poccnet.

The POCC APs will benefit by their connection to Poccnet because they will have access to a large quantity of on-line storage (on DBS) and will have backup equipment available when needed for special operations or to substitute for malfunctioning units. Through the Poccnet IPC subnet, a POCC AP will be able to obtain services required to support its payload which cannot be provided at a reasonable cost by the AP itself. These services include interfaces to NASCOM, Telops, and DEMOS as well as connections to outside communication facilities to provide for remote science operations centers.

2.4.2 DATA BASE STORAGE SYSTEM

Decreasing hardware costs, increasing mission complexity, and the availability of sophisticated data management software have resulted in increased use of data base techniques in control center implementations at GSFC in the last few years. This trend will accelerate in the future as standard, proven data base management systems become available on minicomputers and as system designers and programmers become aware of the advantages of the data base approach.

The DBS subsystem responds to all Poccnet requirements for continuously available on-line storage. It provides standard file and data structure specifications as well as file storage and retrieval services. There will also be provided a DBMS for use by personnel responsible for generating and maintaining payload and system-related data bases. The DBS concept is discussed in paragraph 5.4 and in Appendix 5B.

To make its services continuously available, the DBS subsystem would be implemented as a dual-processor system connected to all other host computers through the IPC subnet. This dual-redundancy system would provide a point of high reliability within Poccnet for backup or restart capability for the Data Operations Control Center (DOCC) system, the IPC Poccnet Message Processors (PMPs), and for the individual

POCCs. The hardware configuration proposed for the Pocnet DBS subsystem consists of two processors connected to tape drives and disk drives through their controllers. Each processor is connected to Pocnet through a separate PMP to ensure high reliability. Both processors are connected to each of the dual-port disk controllers and every disk drive is connected to two controllers. This configuration ensures that one of the two processors will be able to reach data on a given disk in the event of a processor or disk controller failure. The total on-line data storage capacity required in Pocnet is approximately 2 gigabytes. Each processor would also be connected to a tape controller and several tape drives. These tape drives would be used to journal all data base transactions, to move data on or off line, and for data base backup/restore activities.

The dual-processor system is only highly desirable and not absolutely required for the DBS. Early implementations of Pocnet could provide DBS in a single-processor configuration. In fact, DBS need be nothing more than a designated AP; however, such a system would have many operational deficiencies.

To take maximum advantage of currently available data management software but still be able to incorporate new techniques or systems as they are developed, it is proposed that the DBS system provide a layer of interface software between Pocnet users and the data base management system. This will provide full use of the data base approach to the user, while isolating him from the idiosyncracies of a particular vendor's system. This is another example of virtualization of resources provided by the Pocnet approach.

2.4.3 VIRTUAL INTERFACE PROCESSOR

The virtual interface processor (VIP) is a Pocnet host which virtualizes the POCC MOR devices, that is, hides the characteristics of the real devices in the MOR by transforming them into "virtual devices". A virtual device (VD) is created by a Pocnet process designed to control (or simulate) a physical input/output device and to make this physical or simulated device available to other Pocnet processes through IPC. By use of VDs, real physical devices are virtualized, that is, their manufacturer or model-dependent characteristics are hidden from the user. Instead, the user sees an "ideal" device (a VD) and his applications programs can be written without regard to which real devices will be used for input/output at run time. A VIP concept is discussed in paragraph 5.5 and in Appendixes 5A and 5C.

The most general VD is the random-access block display (RABD). Any VD implemented is a special instance of a RABD characterized by a subset of the attributes and capabilities of the RABD.

The physical prototype of the RABD is a CRT display. However, the RABD has attributes and control characteristics which considerably exceed those of existing real devices. In particular, the external physical medium is logically divided into nonoverlapping, named, rectangular subregions. Text outputting and inputting and the majority of VD controls may be performed with reference to one of these regions. Associated with each region is also a system of protection.

User programs use VDs by means of GET, PUT, MODESET, and SENSE commands. GET and PUT commands transfer data to and from the program. MODESET commands are used to alter the internal state of the VD so that subsequent data and commands will be interpreted in a specific way; for example, a user might use MODESET commands to name a region of a RABD and to specify its size or to set tab positions. SENSE commands would be used to request data concerning the internal state of a VD (ready, busy, etc.).

VIPs also support virtual users of virtual devices. A virtual user is a specification of "ideal" characteristics of generalized interactive users of devices. Such time sequence scenarios as menu presentation and selection, conversational system operation, etc., are defined as subsets of the generalized virtual user.

2.4.4 GATEWAYS

Gateways provide for process-to-process communication across network boundaries. In transferring information into and out of Poccnet, the gateway processor must effect a translation between the protocols of Poccnet (at the relevant levels) and the protocols adhered to in the environment of the remote system. These protocols, which vary considerably from one network or remote processor to another, are intended to monitor and control both the validity and the flow of data. Several forms of this communication can be identified:

- Gateway to modem.
- Gateway to local CPU adapter.
- Gateway to local gateway.

The gateway-to-modem communication is the most general since it permits the transfer of data to/from any remote node (computer, user terminal, RJE terminal, data acquisition hardware, and communications controller). However, this form of communication is limited in speed to teleprocessing rates (usually less than 100 kbps and often much less) and may be error-prone owing to the use of voice-grade signal carrier facilities.

The gateway-to-local-CPU-adaptor communication is the most efficient since it permits a very high-speed transfer of data between a local data processing system and a Poccnet process through IPC. However, it requires special, rather expensive equipment and custom software, for each type of local processor requiring a data connection to Poccnet. This type of communication is exemplified by the Network Systems Corporation network adaptor required to interface a Poccnet gateway to the GSFC Integrated Telecommunications Distribution System (ITDS).

The gateway-to-local-gateway communication lies between the first and second types in terms of both speed and generality. An example would be communication between a gateway and a local Arpanet IMP.

In all cases, the goal of the gateway computer is to make a remote system seem like a Poccnet process and make a Poccnet process seem like a terminal of some sort to the remote system. Initially, Poccnet will be interfaced through gateways to Telops, JSC MCC, and the IBM 360 support computers at GSFC. Poccnet gateways are discussed in paragraph 5.6.

2.5 INTER-PROCESS COMMUNICATION SYSTEM

The Inter-Process Communication (IPC) system performs the functions required for two processes executing on different host computers to transfer programs, control information, or data between themselves. Processes located in application processors, VIPs, TACs, or other host computers connected to the IPC subsystem can make use of these functions. The IPC system is discussed in Section 6 of this report.

2.5.1 INTER-PROCESS COMMUNICATION REQUIREMENTS ANALYSIS

The IPC system is functionally the distribution medium for all computer-to-computer communications between host computers. The IPC must provide real-time host communication by supplying logical channels between pairs of communicating

processes running on Pocnet host computers. It must provide the means to set up and disconnect these logical channels quickly and easily. Operationally, the IPC system must be expandable and easy to use and manage, and it must provide performance monitoring. A major performance requirement is that a channel speed of 2 megabits be handled in real-time. This is required to interface the T1 communication channels now available from common data carriers.

2.5.2 DESIGN ALTERNATIVES

A number of design approaches were considered for the IPC system. These were the common access memory design, crossbar design, common bus design, basic loop design, and computer subnet design. The computer subnet approach is recommended because the functional, operational, and performance requirements are more completely fulfilled than in the other approaches. Major requirements satisfied by the computer subnet are as follows:

- a. The failure-effect and failure-reconfiguration characteristics are very good.
- b. The approach offers existing and proven state-of-the-art technology demonstrated by similar networks which have been implemented.
- c. Standard communication protocols are available for node-to-node and process-to-process communications.
- d. Central control of the subnet can be provided by one of the host computers.

Each of the alternative approaches has significant shortcomings in one or more of these areas. In addition, the subnet approach is cost-competitive when commercially available minicomputers and intelligent channels are used for its implementation.

2.5.3 IPC SYSTEM DESCRIPTION

The IPC subnet will utilize digital processors called Pocnet message processors (PMPs). The function of the PMP is to transfer frames through the network, to keep performance statistics, and to detect failures of equipment which are under PMP control. The PMPs are under the supervision of the DOCC (paragraph 6.5). Logical channels will not be constructed between processes without the authorization of the DOCC.

Paragraph 6.4 describes the PMP hardware characteristics, the software organization, the frame queue, the frame routing table, the channel statistics, the timing estimate, and the core estimate.

The DOCC is responsible for allocating computer resources and configuring them into partitions. Logically, the DOCC software can be divided into two main tasks, the IPC DOCC and the host DOCC.

Basically, the IPC DOCC performs low-level tasks such as monitoring physical channel activity and maintaining a record of all logical circuits in the network.

The host DOCC performs higher-level tasks such as maintaining alphanumeric resource directories and handling requests from resources which desire to establish or discontinue a communication path.

Examples of the operator command language and the monitoring capability of the IPC DOCC are discussed in paragraph 6.7.

To initialize the DOCC and the PMPs thus establishing a minimum Poccnet system, the following functions would be performed:

- a. Load the PMP operating system into each PMP locally. Then assign logical names to each PMP and the channels which are directly connected to each PMP.
- b. Load the DOCC operating system into the DOCC and define the cluster configuration. The DOCC must compute routing tables for each PMP and transfer each table to the correct PMP.
- c. Establish a minimum operating environment by having the hosts define themselves and their resources.

User level commands for use of IPC would be sort of a STOL dialect. In general, these user level commands permit the following:

- a. Resources to be defined to the DOCC.
- b. Resources to be connected between two systems which have the need to communicate via a logical channel.
- c. Resources to transfer data over the logical circuit by using sends and receives.
- d. Resources to be disconnected once the communication has been completed.

2.6 APPLICATIONS ENGINEERING

The purpose of Poccnnet is to provide standard POCCs and standard POCC spacecraft control computing support and interfaces. This is done by performing the common systems engineering one time for all GSFC POCCs then developing standard systems for the Model POCC together with common software modules developed under the Poccnnet Applications Engineering activities.

The Model POCC serves the following three basic purposes:

- a. Identifies and describes POCC functional capabilities and requirements.
- b. Acts as a driver for design by identifying candidate software for modularization and standardization.
- c. Serves as a tool for familiarization and planning.

The repository for common software modules is a collection of relevant data pertaining to software identified as being a good candidate for modularization and/or standardization in POCC and POCC support systems development. It is intended to be a prime source of design and implementation ideas for software packages typically found in POCC and support systems.

2.6.1 DBS APPLICATIONS

The DBS is primarily a storage and retrieval facility for data, but it will also offer a small number of basic services related to its primary objective. These services are provided in the form of applications programs which run on the DBS computers or on other applications processors on or off Poccnnet and include the following:

- a. Data, catalog, and library management and maintenance.
- b. Software engineering management and configuration management tools.
- c. Document preparation and maintenance facilities.
- d. Computer conferencing (mailbox services).

Each of these is described in paragraph 7.3 of this report.

2.6.2 SIMULATION

Another important applications area is simulation. There are four categories of project drivers and mission support drivers for simulations: training, operations, analysis, and payload integration and test.

Resource sharing will be maximized within the family of payload simulations by making use of a functional module in as many family members as possible. Reusability of the software components will be enhanced by identifying hardware subsystems to be used on one or more series of spacecraft, carefully developing software models to parallel the hardware operation, and by reflecting the modeling of each spacecraft subsystem in a table rather than by hard-coding the model.

Simulators will be designed to perform both on-line and off-line simulations. The Simulation Director will be able to control and monitor on-line simulation operations from his console. This includes the capabilities for making changes in the simulation parameters, terminating the run, restarting (with or without a new configuration), checkpointing, retesting, and idling.

Simulations will be capable of running in an off-line mode for program development, testing, and analysis. The off-line mode indicates that the command uplink from the POCC, the telemetry downlink to the POCC, or both are disconnected. In this type of operation, simulation(s) of the appropriate link(s) will be available. By appropriately setting a run time parameter, the user may cause the simulation to be run in real-time or in non-real-time.

Simulation requirements and design philosophy are treated in depth in paragraph 7.4 and Appendix 7B of this report.

Just as the requirements for modularity, reusability, and resource sharing are of paramount importance in developing the spacecraft and POCCs for the 1980s, so they are for development of payload and POCC subsystem simulations.

2.6.3 ON-BOARD COMPUTER SUPPORT

An increasingly important area of POCC responsibility is the support of payload on-board computers (OBCs). On-line OBC functions vary from payload to payload. In general, the POCC is responsible for loading the OBC memory, dumping the OBC memory or status buffer, maintaining an image of the current OBC memory contents,

printing OBC dumps, and controlling OBC operations. An important area of OBC support by the POCC is the preparation and uplink of OBC data blocks. The primary objective of off-line OBC support is the preparation of data and program loads for use by the POCC. A typical off-line OBC programming support system consists of the following four major components:

- a. Assembler.
- b. Relocating loader.
- c. OBC simulator.
- e. Executive.

Refer to paragraph 7.5 for a more complete description of the OBC support.

2.6.4 COMMAND MEMORY MANAGEMENT

A command memory management (CMM) system is designed to manage and integrate command requests involved in the creation of memory loads for an on-board memory unit referred to in this section as a stored command processor (SCP). The SCP will sometimes be a special-purpose hardware box in the payload, but, more recently, SCP is implemented as a software function in the OBC. The CMM system will accept, merge, and transform command requests into a spacecraft command format from which spacecraft memory loads will be generated.

In order to accomplish its tasks, the CMM system requires that its data base have available the following detailed information:

- a. The structure and operation of the spacecraft command processor.
- b. The various command formats.
- c. Interface between the CMM facility and the transmitting POCC.
- d. Constraints and limitations imposed on the CMM system by the SCP.
- e. Any special requirements and/or considerations that may be imposed on the CMM system by the project.

CMM is briefly discussed in paragraph 7.6 of this report, and Appendix 7C provides a description of a typical CMM system.

2.7 OPERATIONS

The Poccnnet operations study includes the concepts, O&M functions, and facility requirements for the operation of Poccnnet. Poccnnet operations entail the operation of the Poccnnet system resources and the operational management of the Poccnnet system. Briefly stated this will include the following:

- a. The support interface between Poccnnet and the users. This responsibility will include all operations control functions, including preliminary, real-time or emergency scheduling and resource allocation functions; establishing and verifying configurations; data flow monitoring and accountability; fault isolation, repair, and evaluation; and testing and simulations involving Poccnnet resources.
- b. Providing operations support to users in such areas as developing Poccnnet schedules, documentation changes, information requests, analyzing Poccnnet service performance, the use of the DBS, and software validation and configuration control.
- c. Providing interface coordination between Poccnnet and external support elements such as Orbit Determination, NASCOM, NCC, SMCC, etc.
- d. Providing the maintenance and operation of the DOCC, the DBS systems, and the other Poccnnet system resources. Overall operational management of the Poccnnet system will be provided by the DOCC.

Each of these areas is discussed in Section 8.

2.8 USING POCCNET

A Poccnnet user is defined simply as any system which either inputs data into or receives data from a Poccnnet host. In general, users of Poccnnet will find their interfaces simpler but more flexible than previous POCC interfaces. This is due to the inherent characteristics of a distributed network and the implementation of standard solutions to common requirements. The Poccnnet methods of providing interface solutions are described in Section 9.

Users interface with Poccnnet at different functional levels roughly corresponding to the time before launch and the phase of the user system development process. However, the boundaries of these levels are not rigidly defined, and, in fact, a user may be involved at several levels simultaneously.

2.8.1 OBTAINING INFORMATION ABOUT POCNET

The Pocnet User Handbook and the Pocnet user interface engineer will be the user's sources of information concerning Pocnet. The Pocnet User Handbook will provide the user with most of the information required to interface with Pocnet, and it will reference other documentation where more detailed interface information may be obtained. The handbook will inform the user as to how he may place formal requirements on Pocnet, describe the standard systems and services available, and define operational procedures. Tests and simulations required to integrate the user's unique hardware and software systems into Pocnet and to proof-test the total system will be included. Procedures will be defined to assist in the isolation and correction of the problems encountered during testing or in actual operations.

A Pocnet user interface engineer will be officially assigned to serve as the interface between the user and Pocnet. The interface engineer will provide the user with information about Pocnet and will assist the user in meeting his support needs.

2.8.2 FORMAL REQUIREMENTS

Once a user is familiar with Pocnet and knows what he wants from Pocnet, he is then prepared to place formal requirements upon Pocnet. A Configuration Control Board (CCB) will be established with the authority to approve all requirements on Pocnet systems and any changes to those systems. The CCB will provide the rational configuration management to prevent the proliferation of unique systems and the implementation of systems not conforming to Pocnet standards.

All Pocnet interfaces will be managed and controlled by Interface Control Documents (ICDs). The ICD will spell out the letter and intent of each interface agreement. ICDs will spell out communications protocol, data rates, data formats, electrical and mechanical interfaces, etc. No interface will be allowed with Pocnet which is not controlled by an ICD.

2.8.3 USER OPERATIONS

The user's operations interface with Pocnet will be with the Pocnet DOCC. The DOCC will provide operations support in developing Pocnet schedules, documentation, DBS use, obtaining information, and establishing configurations. The DOCC will provide interface coordination with external support elements and will provide assistance

in planning mission readiness tests and simulations or in obtaining special support for critical support periods. In the event of problems, the DOCC will assist in fault isolation, device or path replacement, and obtaining alternate devices or processes.

SECTION 3
SYSTEMS REQUIREMENTS

SECTION 3

SYSTEMS REQUIREMENTS

CONTENTS

<u>Paragraph</u>		<u>Page</u>
3.1	General	3-1
3.2	External Systems Interfaces	3-1
3.2.1	Shuttle Mission Control Center	3-2
3.2.1.1	Telemetry	3-2
3.2.1.2	Commanding	3-3
3.2.1.3	Operational Information	3-6
3.2.2	NASCOM/STDN/NCC	3-8
3.2.2.1	Introduction	3-8
3.2.2.2	TDRSS Service Capabilities	3-8
3.2.2.3	Telemetry Data Interfaces	3-10
3.2.2.4	Command Data Interfaces	3-11
3.2.2.5	Network Control Center	3-12
3.2.3	Kennedy Space Center	3-18
3.2.4	IUS/SSUS Operations Control Center	3-24
3.2.5	Spacelab POCC	3-29
3.2.6	Operational Support Computing	3-29
3.2.6.1	Orbit Data Operations	3-29
3.2.6.2	Flight Dynamics	3-31
3.2.6.3	Mission Planning and Scheduling	3-32
3.2.7	Data Processing Facilities	3-32
3.2.7.1	Telemetry On-Line Processing System	3-32
3.2.7.2	Image Processing Facility	3-33
3.2.7.3	Spacelab Data Processing Facility	3-34
3.2.8	Data Accountability System	3-34
3.2.9	Mission Operations Center	3-35
3.2.10	External Experimenters	3-35
3.2.11	Miscellaneous	3-36
3.3	Model POCC Requirements	3-36

CONTENTS (Cont)

<u>Paragraph</u>		<u>Page</u>
3.3.1	NASA/GSFC Mission Model	3-37
3.3.2	MMS Model POCC	3-39
3.3.2.1	Introduction	3-39
3.3.2.2	Model POCC Functional Requirements	3-39
3.3.2.3	Model POCC System Control	3-46
3.3.2.4	Model POCC System Improvements	3-49
3.4	Section 3 References	3-51

ILLUSTRATIONS

<u>Figure</u>		<u>Page</u>
3-1	MMS-Type Spacecraft Command Flow	3-5
3-2	Scheduling and Status Block Diagram	3-13
3-3	Desired Levels of Support System Test/Verification Capabilities	3-16
3-4	POCC-to-KSC Interfaces (Composite)	3-19
3-5	POCC-to-KSC Interfaces (Payload at PPF)	3-20
3-6	POCC-to-KSC Interfaces (Payload at O&C)	3-21
3-7	POCC-to-KSC Interfaces (Payload at OPF)	3-22
3-8	POCC-to-KSC Interfaces (Payload at the Pad)	3-23
3-9	Data Links, IUS/Payload Within Orbiter Sphere of Influence	3-26
3-10	Data Links, Before IUS/Payload Detachment (TDRS Not Available)	3-27
3-11	Data Links After Synchronous Orbit Achieved	3-28
3-12	Launch Support Period Chart	3-38

SECTION 3

SYSTEMS REQUIREMENTS

3.1 GENERAL

Pocnet requirements are derived from the fundamental purpose of Pocnet to provide standard POCCs and standard POCC interfaces and support. The standard interfaces are driven by external systems; the standard POCCs are driven by the NASA and GSFC mission models and the nature of standard spacecraft such as MMS.

Paragraph 3.2 identifies the GSFC POCC external interfaces anticipated in the 1980s and describes them in terms of the functional requirements they place on POCCs. How these requirements affect the systems within Pocnet are elaborated upon in Section 4, Systems Engineering.

Paragraph 3.3 gives an estimate of the number of POCCs required at GSFC in the 1980s and presents their anticipated envelope of requirements. Included in paragraph 3.3 are the requirements which an MMS spacecraft would place upon a Pocnet Model POCC, including discussions of the POCC special support computing required.

Paragraph 3.4 provides reference material used in the preparation of this study. The reader is referred to this material at various times throughout this section.

3.2 EXTERNAL SYSTEMS INTERFACES

The following are the GSFC POCC external interfaces anticipated in the 1980s:

- a. Shuttle Mission Control Center (SMCC).
- b. NASCOM/STDN/NCC.
- c. Kennedy Space Center (KSC).
- d. IUS/SSUS OCC.
- e. Spacelab POCC.
- f. Operational Support Computing.
 - (1) Orbit Data Operations.
 - (2) Flight Dynamics.
 - (3) Scheduling.

- g. Data Processing Facilities.
 - (1) Telops.
 - (2) Image Processing Facility (IPF).
 - (3) Spacelab Data Processing Facility (SDPF).
- h. Data Accountability System (DAS).
- i. Mission Operations Center (MOC).
- j. External Experimenters.
- k. Miscellaneous.

The following paragraphs describe these interfaces in terms of the functional requirements they place on POCCs.

3.2.1 SHUTTLE MISSION CONTROL CENTER

The Space Transportation System (STS), or Space Shuttle, flight operations and control will be the responsibility of the Shuttle Mission Control Center (SMCC), which is located at Houston, Texas. This section discusses the functional SMCC/GSFC interface requirements for telemetry, commanding, and operational information.

3.2.1.1 Telemetry. It is the present understanding that the Orbiter telemetry rate will be 96 kbps (on S-band). The Ku-band rate will be 192 kbps. It is GSFC's position that payload data that is in the Orbiter operational instrumentation (OI) link will be sent to SMCC for cleanup prior to transmission (via NASCOM) to Pccnet. A cleaned-up downlink is one in which all the data is in original payload fixed formats and time ordered with respect to the on-board clock within a stated time correlation accuracy (see Reference 1). It is assumed that only payload and payload-related data will be sent to GSFC.

It is assumed that the minimum amount of payload data required for each payload is 1 kbps of actual telemetry and 2 kbps of tape recorder playback; that is, for n payloads, n kbps is the minimum amount of payload data required plus tape recorder data as required by each payload. The maximum amount is TBD.

The following payload and payload-related data is required by the POCC:

- a. Payload data that is in the OI downlink.
- b. Time correlated state vectors on Orbiter orbit and attitude.

- c. Time correlation between the Orbiter clock, the spacecraft clock, and ground time.
- d. An indication of what the downlink format is.
- e. The exact time of free-flyer release.
- f. Other operational information as required (TBD).

Some questions/assumptions are as follows:

- a. Question: How will the POCC be made aware of a change in the Orbiter downlink format?

Assumption: A format indication will be inserted into the data by SMCC as well as being available by voice.

- b. Question: Will data be received from SMCC at a continuous rate?

Assumption: SMCC will transmit payload and payload-related data to Pocc-net in 4800-bit NASCOM blocks on one or more TBD kbps lines.

- c. Question: Will Orbiter orbit and attitude data be put into a buffer at SMCC and be accessed by the POCCs when needed? Similarly for data in the Orbiter payload command buffer.

Assumption: As required, payload-related data (Orbiter orbit and attitude, the Orbiter payload command buffer) will be placed into TBD formatted 4800-bit NASCOM blocks and sent to the POCCs in real-time on the same lines as in the above.

- d. Question: Will payload-related parameters be processed by SMCC?

Assumption: These parameters will be processed by SMCC prior to transmission to the POCCs.

- e. Question: What will be the nature of the payload and payload-related data in the case of more than one spacecraft (up to 5) being deployed from a single Orbiter mission?

Assumption: SMCC will demultiplex payload telemetry and put it into separate blocks; no NASCOM block will contain payload data from more than one spacecraft.

3.2.1.2 Commanding. Each command block will contain a 48-bit preamble word and up to 19 48-bit commands. The block must contain an even number of 48-bit words

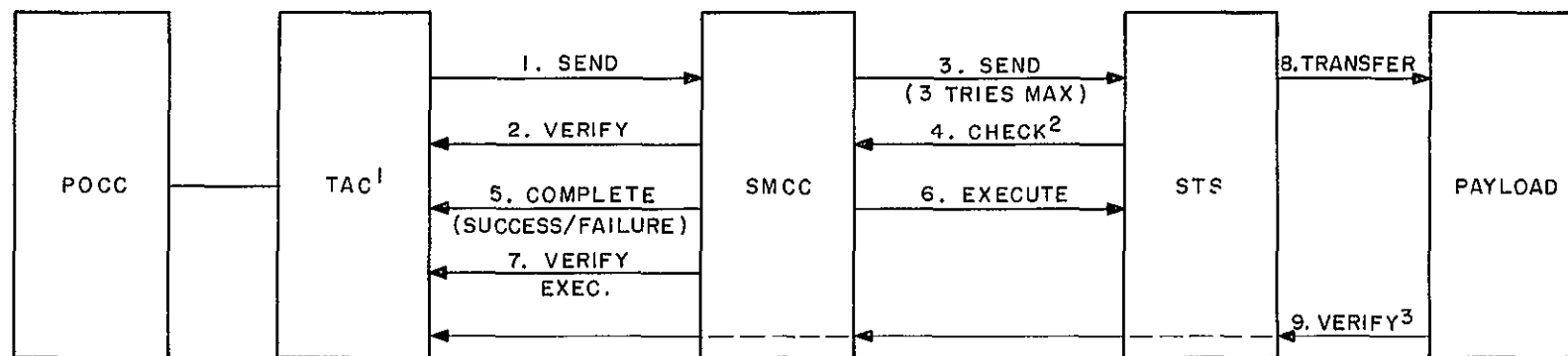
(hence an odd number of commands) and is limited to 20 words (19 commands and a preamble), for a total of 960 command bits per block. See References 1 and 2 for details.

The command flow (send/verify cycle) for an MMS-type spacecraft is shown in Figure 3-1.

Payload commands will be transmitted from the POCC to SMCC via NASCOM. The POCC will provide the command message to NASCOM in a NASCOM block. Each command will be checked at SMCC against a caution list. Note that the Orbiter crew will have some discrete command capability in the Orbiter. There will be voice control from POCC to the Orbiter for the on-board specialist to send commands (Reference 3).

In the realm of command traceability, it is GSFC's position (see References 1 and 2) that, when a block of commands is received at SMCC, an acknowledgement be accomplished via a single message indicating the number of commands received along with an acceptance or rejection of the entire block. Any individual command rejection within a block will require a retransmission of the whole block from the POCC. In addition, the POCC will receive an Orbiter command execute verification (a verification that the command or command block was transferred from the Orbiter payload command buffer to the payload). The final command execute verification will be at the POCC from payload telemetry in the OI data link. The STS operations system will be required to provide traceability while the command exists in the Orbiter command format (from SMCC through the STDN/TDRS, NASCOM, and the Orbiter avionics).

Time factors are important in the commanding cycle. Referring to Figure 3-1, the least defined segments of this cycle are steps 3 and 4, uplinking a command block and verifying it against the payload command buffer in the general-purpose computer (GPC) downlist. There is a minimum of 2 seconds required for steps 2 and 3. Note that if the uplink and downlink bit error rates (BER) are 10^{-4} (Reference 4), then about 1/10 of the uplinked command blocks will arrive at the Orbiter with an error; furthermore, about 1/10 of the downlinked images of the payload command buffer in the GPC downlist used for comparison (step 4) will arrive with an error. GSFC has formulated a requirement of a downlink BER of 10^{-5} (Reference 4), which will improve this condition. Thus, for time-critical commands, time-tagged commands for delayed execution out of the payload command memory will be necessary. Hence, precise time correlation between the spacecraft clock, the Orbiter clock, and ground time is required.



1 TELEMETRY AND COMMAND PROCESSOR

2 COMPARISON OF PAYLOAD COMMAND BUFFER IN GENERAL PURPOSE COMPUTER (GPC) DOWNLIST WITH THE COMMAND BLOCK SENT.

3 VIA TELEMETRY

0026S-1

Figure 3-1. MMS-Type Spacecraft Command Flow

The POCC will require that SMCC maintain a command history for the duration of an Orbiter flight (1). (Refer to paragraph 3.2.1.3.)

Some questions/assumptions are as follows:

- a. Question: What is GSFC's position on retransmitting command blocks that are correctly received by SMCC?

Assumption: It is GSFC's position that, once SMCC correctly receives a command block, it is SMCC's responsibility to uplink the commands within the block.

- b. Question: What are the implications of the multispacecraft case?

Assumption: Each payload operates independently as above. All command blocks, however, will be sent on the same line.

- c. Question: What are the criteria by which successive command blocks may be sent?

Assumption: The POCC can initiate sending another command block when it receives an Orbiter Command Execute Verification.

- d. Question: How many payload command buffers are there? Will Poccnet be required to manage the traffic through the buffer(s)?

Assumption: There will be one payload command buffer. Poccnet will be responsible for queuing commands for the multiple-spacecraft case.

NOTE

Exceptional spacecraft (foreign or GOES-type) will require special interfaces on the Orbiter.

3.2.1.3 Operational Information. The following is operational information for external interfaces:

- a. Data bases — The following three types of remote data base access capability are required operationally:

- (1) A remote terminal is required to the Mission Support Requirements System in order to access the long-range planning data base.

- (2) A remote terminal is required to Johnson Space Center (JSC) in order to access their operational data base.
 - (3) A remote terminal at JSC is also required in order to access Poccnet Mission Planning System.
- b. Loop tests — The following three levels of loop testing are required operationally:
 - (1) Line working? (POCC/NASCOM loop).
 - (2) Ready for commands? (POCC/SMCC loop).
 - (3) NO-OP to payload? (POCC/payload loop).

NOTE

The purpose of the POCC/SMCC loop is to ensure that SMCC will uplink commands if they are sent. The POCC/payload loop can be fully tested by a NO-OP command. Note that this will include a test of the POCC/STS loop. (See Figure 3-1.)

- c. Monitoring JSC/POCC operations — The following two types of remote computer terminals are required at each site to monitor the overall operation:
 - (1) Up to five CRTs (24 x 80 characters).
 - (2) One line printer (200 lines per minute).
- d. Command history — Requirements for command history comprise the following:
 - (1) Up to 200 most recent payload commands.
 - (2) All commands after mission completion.

There are up to 200 payload commands in the SMCC payload command history buffer. A request by GSFC to dump this buffer on a data link to GSFC in real-time may be made any time during the mission. The data will remain available with a 1-hour turnaround until 48 hours after Orbiter touchdown. In any case, all commands will be archived on magnetic tape.

3.2.2 NASCOM/STDN/NCC

3.2.2.1 Introduction. The Spaceflight Tracking and Data Network (STDN) is a world-wide complex of stations used primarily in support of NASA earth-orbiting scientific and applications satellites and manned spaceflight programs. STDN presently employs a variety of telemetry and command systems operating at various frequencies. By the end of 1980, the Tracking and Data Relay Satellite System (TDRSS) will become fully operational and will be included as part of STDN. TDRSS will consist of two Tracking and Data Relay Satellites (TDRS) operating at geosynchronous altitude (with a third placed between these two to serve as a spare) and a TDRS ground station. The TDRSS era will be characterized by the bent-pipe concept of telecommunications, which will be discussed later.

It is planned that the 1980 Ground Spaceflight Tracking and Data Network (GSTDN) will consist of five orbital support stations, each having a minimum of two S-band links, two launch support stations, and a Network Test and Training Facility at GSFC. (For more detail, refer to paragraph 2.2.2 of Reference 4.) Existing modes of operation will be continued for projects operational before modification of the network. Since Poccnet will not be operational before 1980, it will not have any special capability for the STDN transitional period; Poccnet will be fully ready for the STDN configuration as it exists when Poccnet becomes operational.

In general, the TDRSS and the GSTDN network will service two classes of users: the low earth orbiters (below 12,000 km) by TDRSS and the free-flyers with orbits above 12,000 km (geosynchronous and highly elliptical orbiters) by GSTDN.

The NASA Communications Network (NASCOM) is a global system established and operated by NASA to provide operational communications support of all NASA projects. NASCOM provides voice and data links between GSFC and the STDN stations, SMCC, KSC, etc.

The Network Control Center (NCC) will be located at GSFC and will be responsible for the overall NASCOM/STDN network operational management.

3.2.2.2 TDRSS Service Capabilities. There will be multiple-access (MA) and single-access (SA) service capabilities. The MA service operates on S-band and provides downlink service for 20 spacecraft simultaneously whose telemetry rates do not exceed 50 kbps (References 4 and 5). The MA forward link will be time shared

with a maximum data rate of 10 kbps. There is exactly one MA forward link per TDRS and each forward link can support one user at a time. There is a 15-second switch time between two users of an MA uplink. Thus, accurate scheduling is important, but there will be emergency scheduling provisions in the system.

There are two data channels per MA return link. This means, for example, that real-time telemetry and memory dump (or real-time housekeeping and science data stream) can be simultaneously, independently, and asynchronously transmitted. POCC will not be responsible for demultiplexing these streams; they will appear on two ports. Note that the sum of the data rates in each of the two channels of a link cannot exceed the total for that link. However, if the data rates of the two channels are unequal, the ratio of the power division between them cannot exceed 4:1 irrespective of the data rates on the two channels (Reference 3).

The SA service provides high data rate links. It will be a dedicated service utilized on a priority scheduled basis. Forward link service is provided at S-band (up to 300 kbps) and Ku-band (up to 25 Mbps); return link service is provided at S-band (up to 6 Mbps) and Ku-band (up to 300 Mbps). The following acronyms are employed: S-band SA (SSA) and Ku-band SA (KSA). Each TDRS will have two SSA and two KSA links, both forward and return, that is, two operational TDRSs will provide four SSA and four KSA links. However, since the Ku- and S-band links share a single antenna, simultaneous use can occur only when the signal sources are in the common beam-width (Reference 4). Each return link can consist of up to four data channels.

Any mission that is compatible with the MA system can receive forward or return link support from either the MA or SSA systems. Thus, continuous MA support of a real-time return link and periodic support of a high data rate (experiment) is a viable operational mode for these systems (Reference 5).

Convolutional coding is required on all MA return links; it is optional on the SA return links (Reference 5).

Both the SA and MA services provide tracking data with accuracy comparable to those currently available from the GSTDN stations (Reference 4).

A more complete description of these services appears in References 3, 4, and 5.

3.2.2.3 Telemetry Data Interfaces. The TDRSS communications configuration consists of a full-duplex service between the TDRSS ground station, JSC, and GSFC. The baseline service bandwidth is 1.544 Mbps between the TDRSS ground station and GSFC. During the 1980s, NASCOM plans to supply a minimum of one duplex 56-kbps circuit to each of the GSTDN orbital support stations (Reference 4). Additional bandwidth will be provided when it becomes necessary in support of future approved mission requirements. Poccnet Telemetry and Command Systems (TAC) presently being procured will be able to accommodate average telemetry data rates of up to 600 kbps.

Any multiplexing and demultiplexing of data by NASCOM will be transparent to the POCC. For example, if spacecraft telemetry is at the rate of 64 kbps, NASCOM will distribute the telemetry over more than one 56-kbps channel and will then resequence the data and present it to the POCC as a single stream.

TDRSS will supply only a real-time throughput mode of operation (as opposed to the current store and forward techniques). There will be no STDN-supplied storage capabilities at the TDRSS ground station and no retransmission of lost blocks, except that, at the TDRSS ground station, NASCOM will provide a 30-minute contingent recording capability at a rate of 1.544 Mbps for protecting against possible loss of critical data due to a leased service outage (paragraph 5.5, Reference 5). There will be no protocol for acknowledging blocks transmitted or received.

The TDRSS ground station will not process user spacecraft telemetry data. The bit-contiguous user data will be partitioned into groups of data bits, and a single group will be inserted into the data field of a 4800-bit NASCOM block. No frame synchronization detection of the user spacecraft data frame will be performed. There will be no correlation between the frame sync word of the spacecraft data frame and its location within the data field of the NASCOM block. The data within the blocks can be output to Poccnet in either a restructured spacecraft bit-contiguous (serial) data stream or the block format used for ground transmission; however, if a POCC desires time-tagged telemetry data, it must accept the data in blocked format (refer to Reference 5). This time tag will not be correlated to any bit location in the spacecraft telemetry frame but rather will be the time of receipt at the TDRSS contractor/NASCOM interface of the first bit in the data field of the NASCOM block.

The preceding paragraph applies to data rates not exceeding 1 Mbps. Since the TDRSS is capable of providing a return service of up to 6 Mbps on S-band and up to 300 Mbps on Ku-band, demands will probably be placed upon NASCOM to handle multimegabit data streams. However, at this point in time, NASCOM has no system design for higher data rates, because hard requirements of such have not as yet been established.

The following three problem areas require consideration:

- a. NASCOM intends to fill a block with telemetry data before releasing it. Normally, this could result in several seconds of waiting for data in the case of low telemetry rates. To alleviate this problem, NASCOM has negotiated a time-out parameter of 1 second for individual users.
- b. The time gap between blocks is not necessarily constant (that is, data transmission is asynchronous). The effect of asynchronous data transmission upon input buffers and data handling functions needs further study.
- c. NASCOM should be capable of providing the same data to multiple-access ports.

It is planned that all GSTDN stations will ultimately operate in a TDRSS-compatible bent-pipe mode (Reference 4). Present plans call for GSTDN to phase over to bent-pipe operations by January 1983. If spacecraft data is convolutionally encoded, it will be decoded at the TDRSS ground station.——

Although the NCC will sample the telemetry stream to determine service accountability and to correct any problems, the POCC and other users will also be required to evaluate data and to notify the NCC of any problems.

3.2.2.4 Command Data Interfaces. It is anticipated that all commanding will be open loop (paragraph 4.3, Reference 5); a POCC will input its block-formatted command data to NASCOM, which will route the blocks in real-time to the GSTDN or TDRSS. At this point, if a block contains polynomial encoding errors, it will be rejected. In the absence of errors, the commands will be extracted and immediately sent on to the spacecraft. The blocked data originated by a POCC will have a spacecraft transfer rate which the POCC will control. POCCs will use their own internal time standard to control the rate at which commands are delivered to the NASCOM MDM equipment and command generator in order not to exceed the NASCOM buffer

capacity. There will be a command echo block returned to the POCC by NASCOM to assist in this process. This command echo block will be filled as commands are transmitted from the multiplex/demultiplex (MDM) equipment to TDRSS. When the echo block is filled or if a 1-second time-out occurs, the block is transmitted to the POCC.

The POCCs will not depend upon the command echo block to perform the command function. This command echo block is a requirement specified in Reference 6. Since the POCC is responsible for command transmission, the command echo block will go to the POCC first priority and to the NCC for fault isolation if the NCC requests this service. The prime means for command verification would be derived by the POCC from real-time spacecraft telemetry.

The bit rate clock for transfer of command blocks will be provided to Poccnet by NASCOM (refer to paragraph 5.4 of Reference 5) and will be in the range of 56 kbps to 1.544 Mbps.

Backup command capability using prestored commands and command history capabilities are not planned for TDRSS.

3.2.2.5 Network Control Center. The NCC will provide the following support functions:

- a. Real-time and emergency scheduling.
- b. Data monitoring and service accountability.
- c. Testing and simulations involving Network resources.
- d. Fault isolation.
- e. Ground configuration commands.

3.2.2.5.1 Network Scheduling. The major scheduling activities are shown in Figure 3-2. Users are expected to submit requests for support to the NCC. The NCC believes that a conflict-free schedule can be made for a week's activities. There will always be provisions for spacecraft emergency support. All users will have access to the scheduling data base.

A computer-to-computer interface will be required between the POCCs (Poccnet) and the NCC scheduling system for the transfer of support requirements into the scheduling system (generic and specific) and for the return of schedules and data base information from the scheduling system.

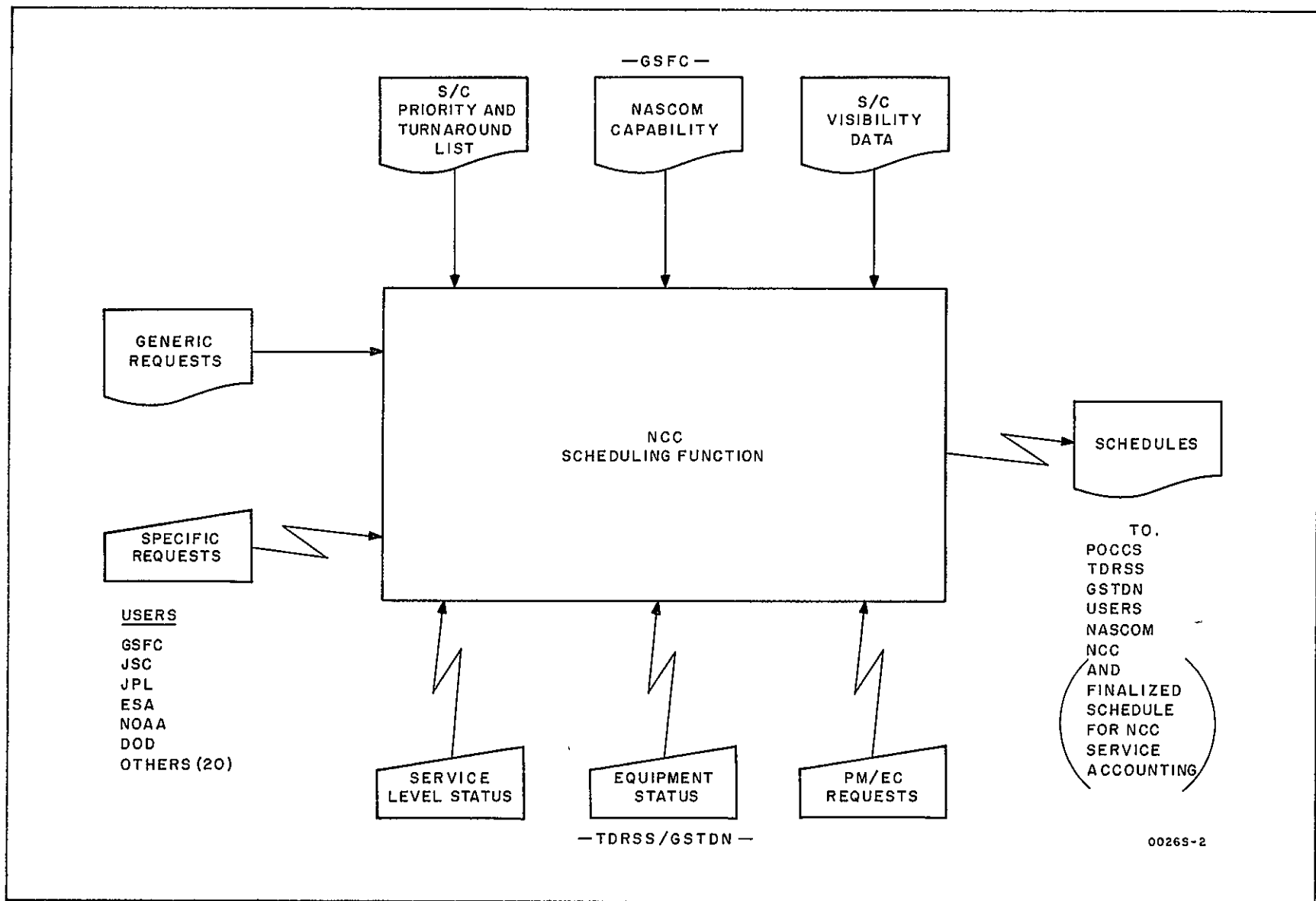


Figure 3-2. Scheduling and Status Block Diagram

3.2.2.5.2 Service Accountability. The POCC will provide the NCC with a postpass quality summary. It is desirable that the performance data be transmitted via inter-computer communications. In addition, the NCC will sample telemetry data in order to monitor performance. The POCC will also notify the NCC of any serious performance problems.

In the event of a common-carrier leased-service outage, NASCOM will establish services on the alternate leased service on a preassigned priority basis (paragraph 5.8 of Reference 5). Users return data which cannot be accommodated on the alternate service will be recorded on the 30-minute contingent recording capability at the NASCOM terminal system at the TDRSS ground station and GSTDN stations. Contingent recording will occur only during the outage.

3.2.2.5.3 Testing and Simulations. (Refer to paragraphs 3.3 and 3.4.2 of Reference 4 and also Reference 7.) The conversion to the throughput mode of operations will dictate greater emphasis on the concept of testing all elements as an operational entity. Testing during the 1980—1990 time frame will fall into the following categories:

- a. Permission testing, which will include compatibility testing to ensure that the proper operational interfaces exist between NASCOM resources, spacecraft control support computing, spacecraft, Poccnet and POCCs; data flow testing; and simulations.
- b. Mission readiness testing, which will include preparation for launch operations, validation of the integrity of Network/user interfaces, and validation of ground station configuration.
- c. Prepass readiness testing.
- d. Performance of fault/problem isolation.

The advent of the TDRSS will necessitate a considerable modification of compatibility testing techniques. Even before the system becomes operational, it will be necessary to simulate relay satellite operations in testing spacecraft which will eventually be designed to be compatible with the TDRSS. This will be achieved by using a test van to simulate both a TDRS and its ground terminal. Once the TDRSS has become operational, either an operational TDRS or the spare in-flight satellite will be scheduled for compatibility testing. In either case, both forward and return link relays between the spacecraft and the user facilities will be verified.

Once signal and format compatibility between spacecraft, Network, and user facilities have been established and intrafacility tests have been performed, data flow tests will be accomplished through as much of the total system as is possible. Pre-mission testing will include the checkout of all required services included in the complete end-to-end support configuration. Therefore, telemetry, command, and tracking data flows will be conducted which will incorporate all the required Network equipment, NASCOM, POCCs, spacecraft control support computing, sensor data processing facilities, and, when possible, the principal experimenters.

Prior to interfacing with a POCC for simulations and tests, initial end-to-end check-out of each participating STDN station (GSTDN and TDRSS) will be accomplished by the STDN by the use of an on-site simulation system to represent the spacecraft and the use of the NCC simulation system to simulate a POCC. A final test of each station/POCC end-to-end configuration will be run as part of the mission readiness test.

For simulations with the TDRS, a small antenna at GSFC and associated simulation hardware in the NCC will be used to represent both links for a user spacecraft. User spacecraft telemetry signals will be transmitted to either the Eastern or the in-flight spare TDRS and relayed through the TDRSS ground terminal back to either a simulated POCC or the actual POCC over NASCOM lines. The forward link will be simulated by transmitting command data through the TDRSS ground terminal and recovering the command data at the NCC or POCC through the small RF antenna at GSFC. The POCC will be able to send commands to the TDRSS on-site simulator (representing the user spacecraft) via the TDRS. The simulator will then respond by outputting user spacecraft telemetry data through the test antenna to the TDRS and proceeding through normal data flow paths from there to the POCC.

Data flow testing and simulations performed as required during the mission phase will be essentially similar to those described in the previous paragraphs.

Figure 3-3 shows the levels of support system test and verification.

3.2.2.5.4 Fault Isolation. The POCC will immediately notify the NCC of any serious performance problem and participate in coordinating the fault isolation necessary to identify the location of the problem. The NCC will coordinate any STDN (GSTDN, TDRSS, and NASCOM) efforts. In the event of a fault that degrades support, the NCC should coordinate with the POCC prior to removing support; that is, the POCC may prefer to continue in a degraded mode rather than undergoing the complete outage caused by service replacement.

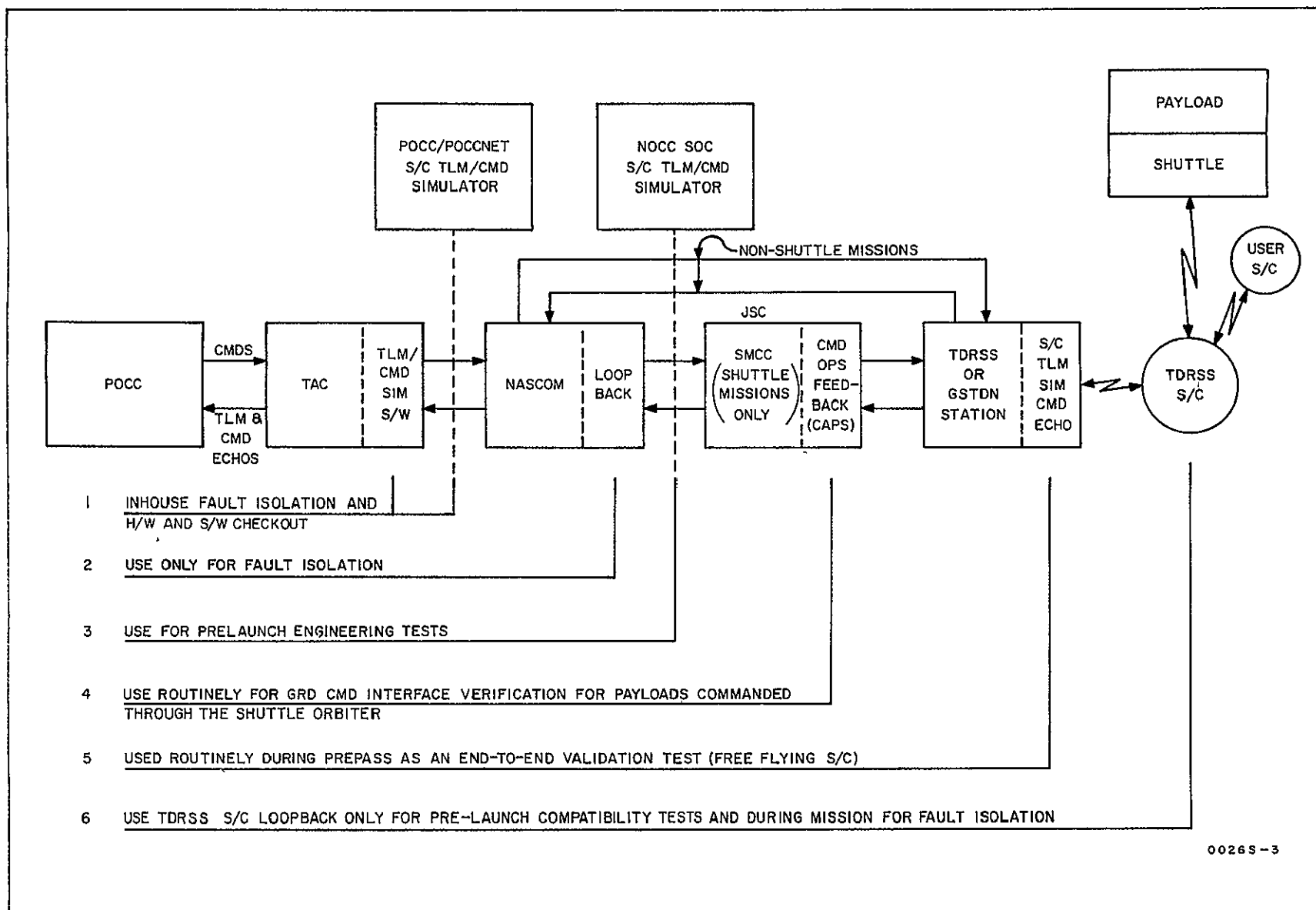


Figure 3-3. Desired Levels of Support System Test/Verification Capabilities

3.2.2.5.5 Ground Configuration Commands. The POCC will be required to send ground configuration command requests to the NCC for changes in the TDRSS services (alternate data streams from spacecraft, change in bit rate, etc.). The POCC will also be required to send ground commands to NASCOM to change the modes of the allocated service (to change the bit rates for commanding and to obtain replay of contingency recorded data). These latter commands can also be sent to the NCC for its information.

3.2.2.5.6 Data Interfaces. The POCC/NCC data interfaces are as follows:

- a. From POCC to NCC.
 - (1) Schedule requests (generic or specific).
 - (2) Ground system directives.
 - (3) Support quality evaluation report.
 - (4) Data base access for resource characteristics, availability, etc.
 - (5) Simulation data (telemetry and command).
- b. From NCC to POCC.
 - (1) Schedule of assigned resources.
 - (2) Performance data (particularly for TDRSS).
 - (3) Data base access for resource characteristics, availability, etc.
 - (4) Simulation data (telemetry and command).

3.2.2.5.7 Problem Areas. The following areas require further consideration:

- a. The NCC scheduling process is not well understood. The generic versus specific schedule requests concept must be clarified (for example, what is the exact nature of generic requests? How much information must the POCC put in the schedule request?). There is disagreement and confusion concerning the STDN schedule timeline. The mechanism for resolving scheduling conflicts is not understood.
- b. The mechanism for mutual POCC/NCC data base access needs to be defined.

- c. The nature of the performance data required by POCCs and the quality evaluation data required by NCC need to be defined.
- d. System fault isolation procedures need to be defined.
- e. The simulation and data monitoring requirements of POCCs, NCC, TDRSS, and GSTDN need to be explored for possible overlap.
- f. The ground configuration command interfaces need to be defined.

3.2.3 KENNEDY SPACE CENTER

For expendable launch vehicles such as Delta and Scout, the launch operations at KSC, Wallops, and Western Test Range (WTR) are ongoing and well defined. This section describes the KSC/STS/POCC interface requirements at a functional level as they are believed to be at the present time. The Launch Processing System (LPS) is not addressed here because its planned use by payloads is not presently known; this does not rule out the possibility of a need to use the LPS, for example, as a possible data base link with the LPS.

Refer to Figures 3-4 through 3-8 during the following discussion. Figure 3-4 is a composite of all interface phases at KSC and is supported by Figures 3-5 through 3-8 which show the interface configuration at the various phases.

When the payload arrives at the Kennedy Space Center (KSC), it is checked out at the Payload Processing Facility (PPF) using the Payload Ground Station (PGS) located there. Payload personnel at KSC will be responsible for payload testing and check-out. The POCC will be checked out using real spacecraft data. Also, the POCC could serve as a backup to the PGS. Thus, the POCC will be able to monitor and command the payload throughout all phases at KSC. See Figure 3-5 for the data flow while the payload is at the PPF.

After payload checkout and flight preparations are completed, the payload is transferred to either the Operations and Control (O&C) building for horizontal integration or the IUS Processing Facility (IPF) for vertical integration. Each of these facilities has cargo interface test equipment (CITE), which is essentially a cargo bay simulator. Here, payload-to-Orbiter interfaces will be verified by CITE. At CITE, there will be a requirement for simultaneous integration of several payloads. The POCC has a

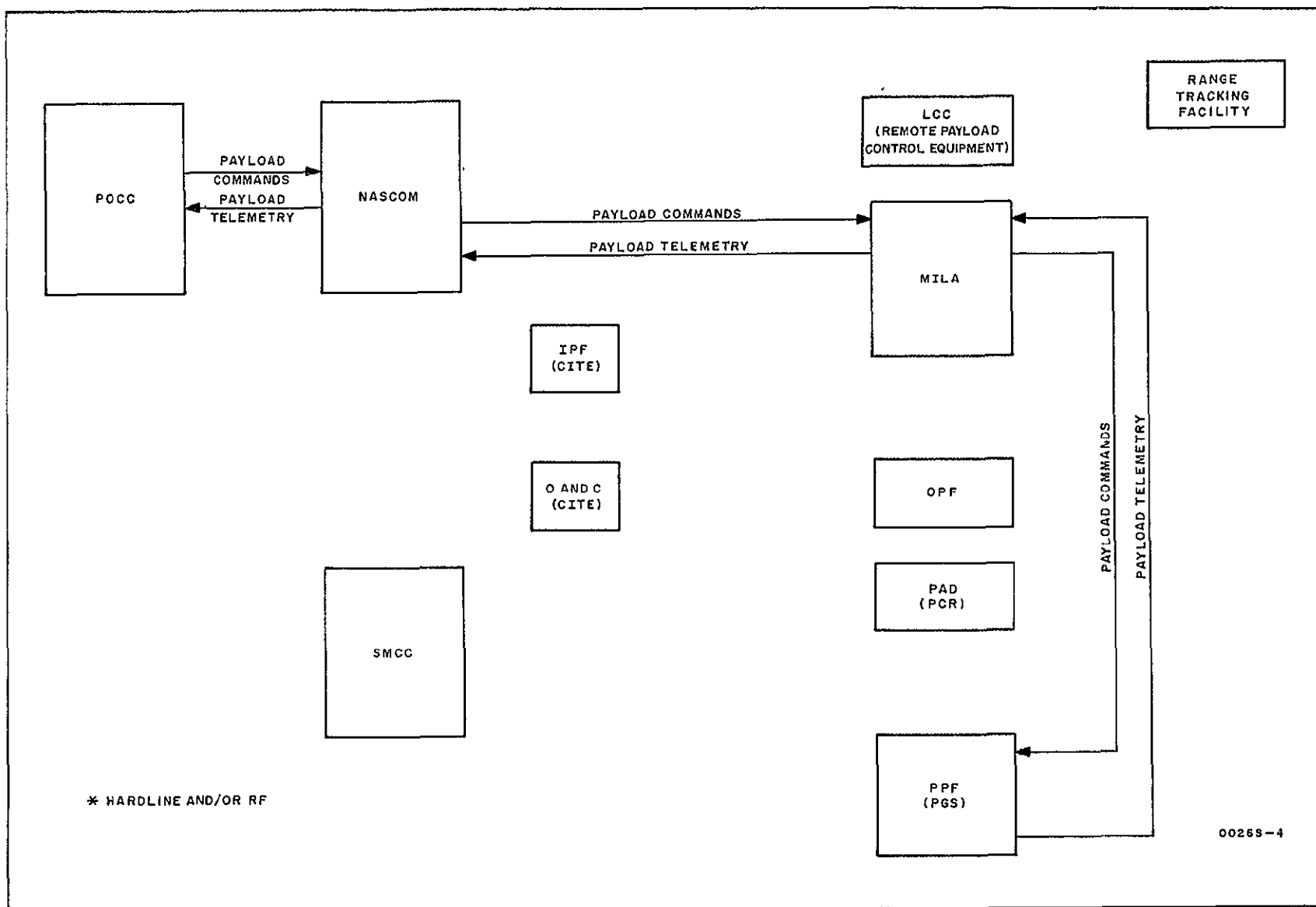
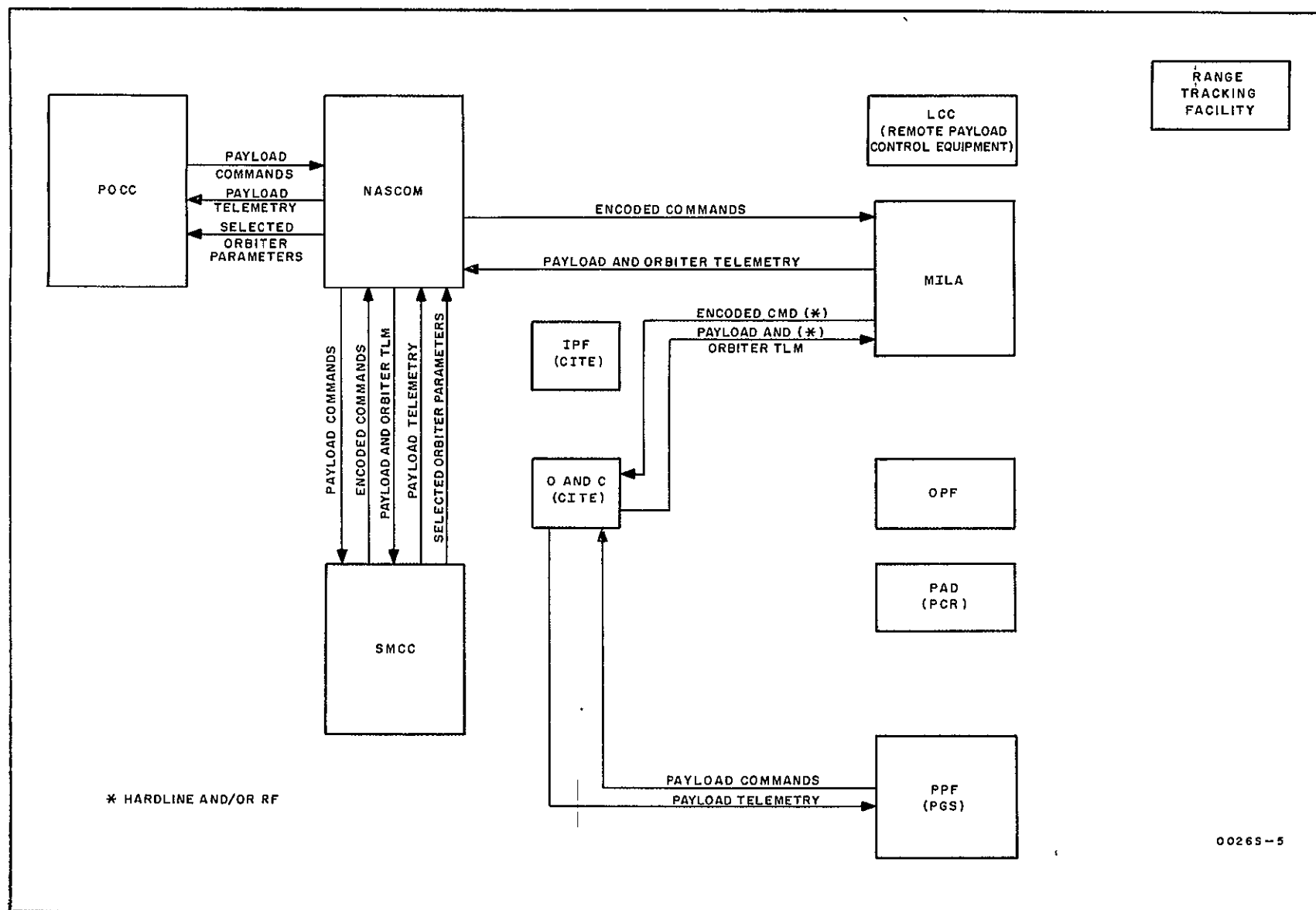


Figure 3-4. POCC-to-KSC Interfaces (Composite)



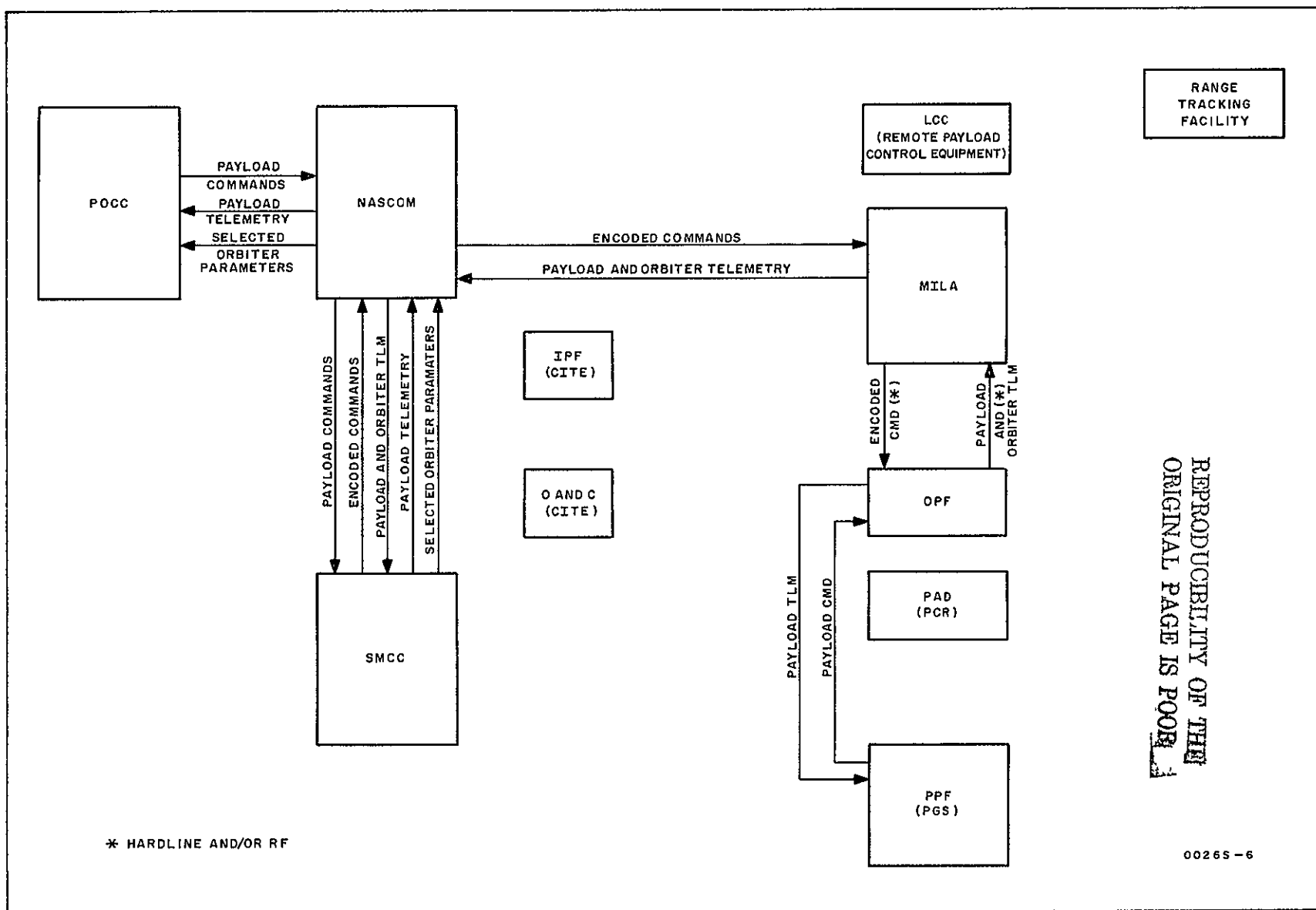


Figure 3-6. POCC-to-KSC Interfaces (Payload at O&C)

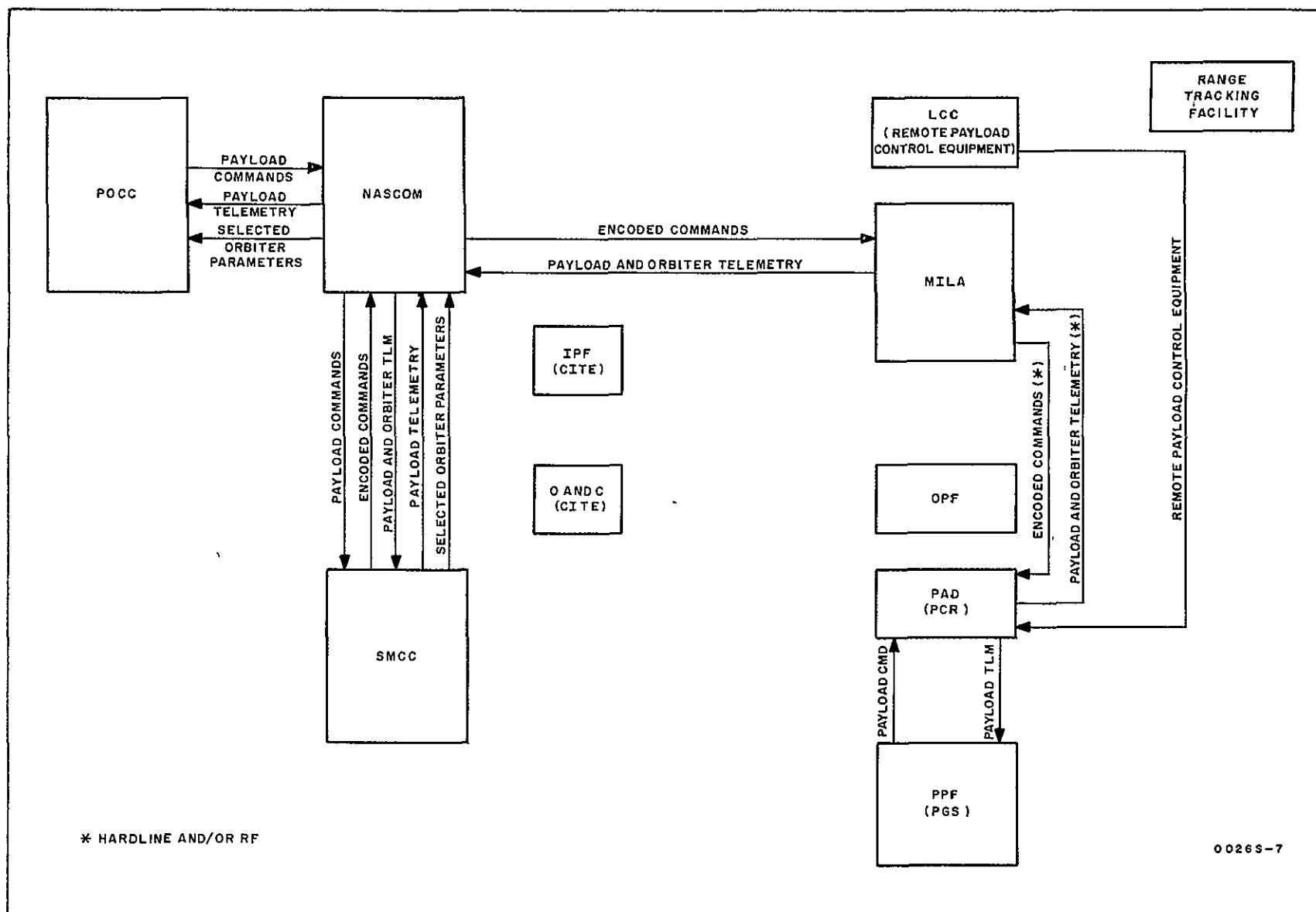


Figure 3-7. POCC-to-KSC Interfaces (Payload at OPF)

requirement to test the Orbiter/SMCC/POCC links. (See Figure 3-6 for the links in the horizontal integration case.) The links are similar for vertically integrated payloads (see Figure 3-4). The PGS will be receiving real spacecraft data via umbilical or some other device.

As seen in Figure 3-4 through 3-8, the POCC will be interfacing with KSC by means of the NASCOM/Pocnet interface (refer to paragraph 3.2.2) and the SMCC/Pocnet interface (refer to paragraph 3.2.1). As such, the POCC expects to transmit and receive only blocked data.

Horizontal payloads are mated to the Orbiter, and interface testing is completed (with possibly several payloads simultaneously) at the Orbiter Processing Facility (OPF). At this time, there will be the first end-to-end data check using real Orbiter avionics. (See Figure 3-7.)

Payload data flow will be essentially the same at the pad as it was in the OPF. (See Figure 3-8.) For payloads installed at the OPF most of the interface testing will be performed at the OPF; however, for pad-installed payloads, interface testing will be performed following installation. If there is a mix of horizontal and vertical payloads, a further end-to-end data check will be required because multiple telemetry streams will be embedded in the Orbiter OI link. Most of the launch pad activity after installation and prior to ascent should consist mainly of payload monitoring. However, there will be some aliveness testing at this time. —

3.2.4 IUS/SSUS OPERATIONS CONTROL CENTER

In order to deliver payloads to trajectories that are outside the Orbiter capabilities (for example, into a geosynchronous orbit) an additional propulsive system is required. The Interim Upper Stage (IUS), the first such system, is being developed by the Air Force. It will be capable of delivering a 1500—5000-pound payload to geosynchronous orbit as well as to other planets. The IUS is three-axis stabilized and has three on-board computers. A study is presently underway which will define the GSFC requirements for the IUS.

Also planned is the Spinning Solid Upper Stage (SSUS). The SSUS-D will be capable of delivering a 2000-pound payload into geosynchronous orbit. The SSUS-A (not yet firm) will be capable of delivering a 4000-pound payload into geosynchronous orbit.

Since both the IUS and SSUS are expendable vehicles, a space Tug may be developed by NASA in the future to provide a completely reusable vehicle.

The IUS will be tracked by GSTDN when possible and may also be TDRSS compatible. The IUS can be autonomous, not needing any ground support. The separation pyrotechnic firing power will be provided by the IUS with a backup firing pulse provided by the payload. The IUS and payload will provide separation signals to verify successful separation (Reference 1).

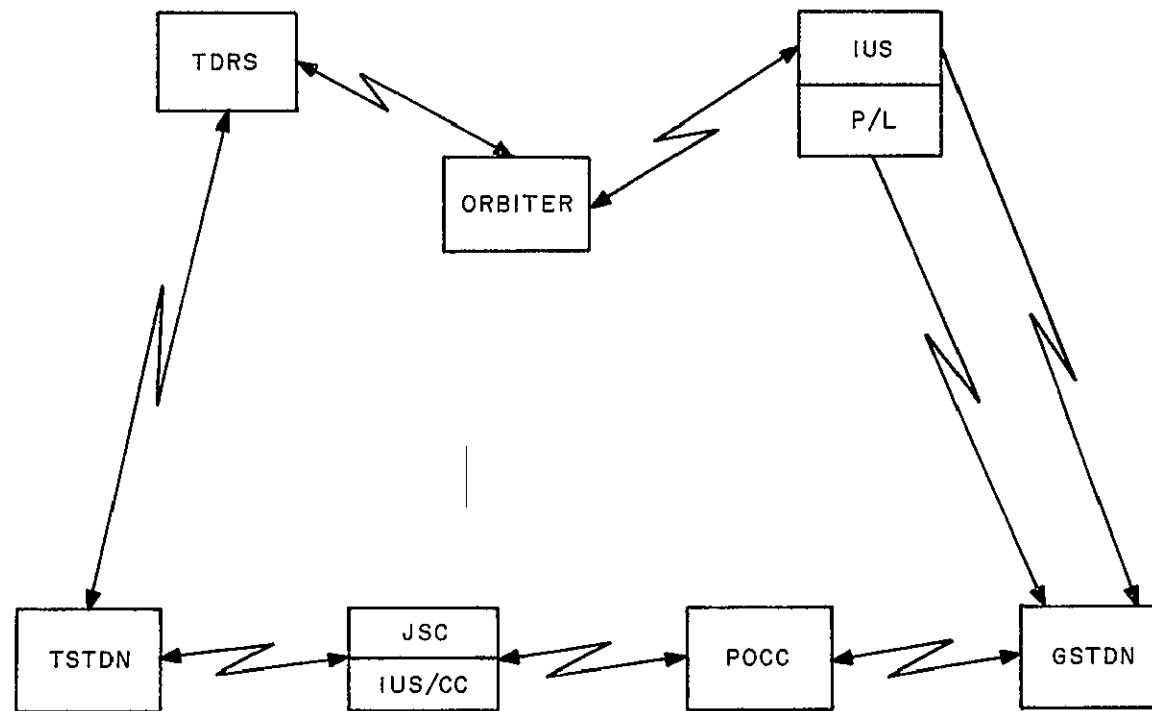
The Orbiter will point the SSUS, and subsequent SSUS action will be autonomous. NASA will have the responsibility of tracking the SSUS and providing attitude and orbit determination. There will be an SSUS-to-Orbiter telemetry downlink only.

Marshall Space Flight Center will coordinate NASA IUS requirements to the Air Force. Figures 3-9 through 3-11 show possible data links from the time that the IUS/payload is within the Orbiter sphere of influence until synchronous orbit is achieved.

The POCC will monitor all payload housekeeping activities during the entire IUS operation provided that a GSTDN station is available (telemetry data will be received whenever the IUS/payload is over a GSTDN station). During all phases of the IUS operation, the IUS Control Center will transmit in near-real-time, through NASCOM to the POCC, IUS motor ignition times, orbit and attitude ephemeris, and prediction data for payload mission evaluation and planning (Reference 1). The POCC should not have to transmit commands to the payload during this phase except in a payload emergency.

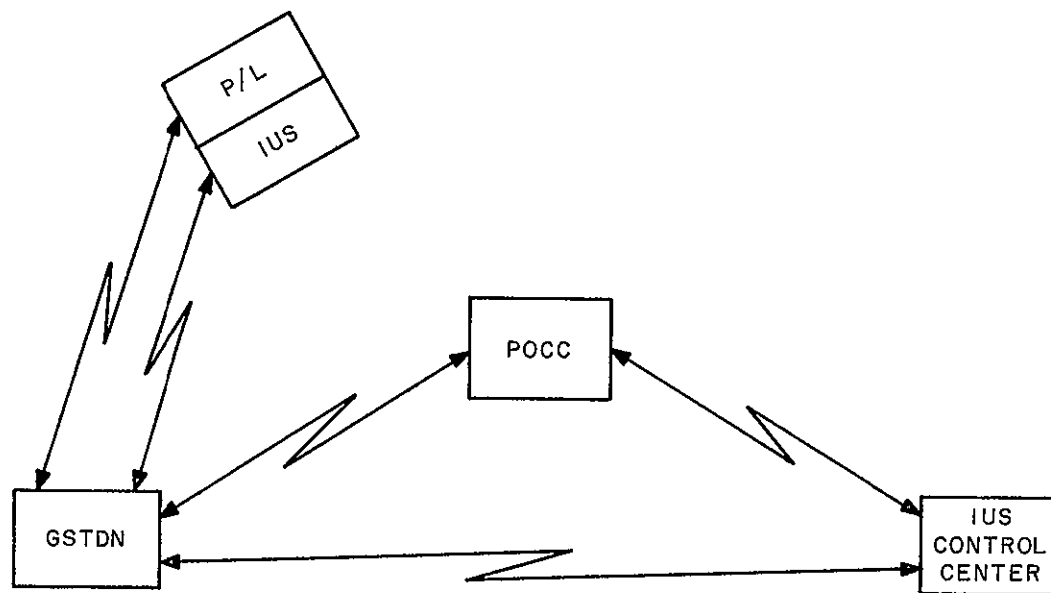
In summary, the assumed data interfaces are as follows:

- a. The IUS Control Center will be supplied with the orbit and attitude of the IUS/payload prior to the first IUS motor burn and those orbital parameters which define the desired geosynchronous orbit (altitude, latitude, and longitude, etc.).
- b. The IUS Control Center will be required to provide the following data in near-real-time:
 - (1) IUS motor ignition times.
 - (2) Orbit and attitude ephemeris and trajectory prediction.



0026S-9

Figure 3-9. Data Links, IUS/Payload Within Orbiter Sphere of Influence



0026S - 10

Figure 3-10. Data Links, Before IUS/Payload Detachment (TDRS Not Available)

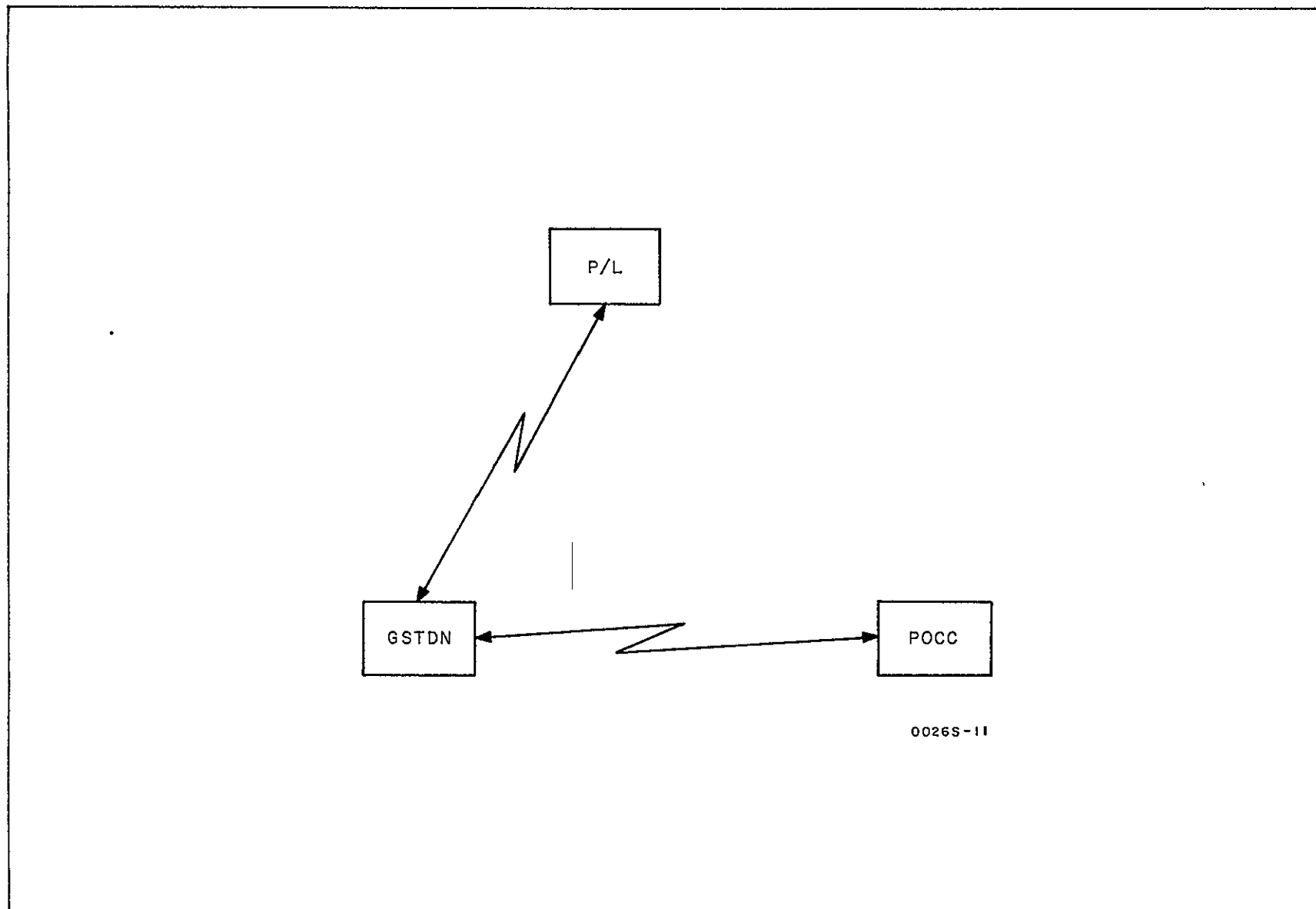


Figure 3-11. Data Links After Synchronous Orbit Achieved

3.2.5 SPACELAB POCC

The Poccnet interfaces required to support experimentation on Spacelab are unknown at this time. The Spacelab POCC will be located at JSC. It is likely that several ASP, AMPS and/or EVAL experimenters will require the capability to support simulations, training, and/or operations from GSFC, since GSFC is providing experiment integration facilities for these payloads. In support of this capability, an interface will be required between the Spacelab POCC and the experimenters, via Poccnet, for transmitting relevant Orbiter and Spacelab data, orbit and attitude information, and instrument housekeeping and quick-look analysis outputs. These latter outputs would normally be available at the POCC, especially if this location were directly supported by the experimenters; however, this information could be derived directly from the instrument output obtained at GSFC via TDRSS. In addition, command capability in real-time will be required between the experimenter and the Spacelab POCC with voice required to support this operation. These interfaces are applicable for simulations, training, and mission operations, where, in the former two cases, all data transferred from the Spacelab POCC to the experimenters will be simulated data.

3.2.6 OPERATIONAL SUPPORT COMPUTING

The external operational support computing required by Poccnet falls into the following three categories:

- a. Orbit data operations.
- b. Flight dynamics.
- c. Mission planning and scheduling aids.

3.2.6.1 Orbit Data Operations. Orbit data is required by POCCs mainly for three purposes. First, the predicted orbit is required to set up operations and configurations which are dependent on spacecraft location, for example, the predicted trace of a spacecraft pointing arc on ATS-6. Second, the predicted orbit is required during real-time operations to properly monitor position-dependent activity such as instrument pointing or operations with TDRSS. Finally, orbit data is required in the POCC for sending to the payload on-board computers (OBC).

In order to access orbit ephemeris/OBC orbit loads/vector data, the POCCs have the following requirements:

- a. For a read-only mechanism into the Orbit Data Operations data base.
- b. For a two-way signaling and data transfer capability, the details of which are TBD.

Orbit Data Operations will provide the following three types of programs:

- a. Standard programs which will be run as routine production and from which the POCCs will periodically receive the output.
- b. Standard programs which will run on request from the POCC.
- c. Special programs which will run when requested to satisfy special mission planning or mission support requirements.

There is a requirement to be able to request the operation of programs on other than a preplanned basis.

Orbit Data Operations will require some stripped telemetry for navigational analysis and OBC orbit load verification. Possible types of such data needed include launch vehicle IGS data, spacecraft accelerometer data, GPS data, orbiter ephemeris, OBC orbit data, and landmark data. Some of the data may be transferred in real-time (such as LV IGS) which implies a need for a signaling capability; some of the data, in non-real-time (such as GPS data). For non-real-time transfers, data will be continually collected and stored and Orbit Data Operations will access it when needed.

In order for the Orbit Data Operations to input schedule requests to the POCC for metric data, a read/write capability is required. Orbit Data Operations needs to write to the POCC schedule-request data base, and it needs to be able to read a POCC request to the NCC and the NCC reply to the request.

At the present time, transfer-of-data delays are from 15 minutes to one or more days. However, there are envisioned requirements of approximately a minute to a few minutes for transfers, depending upon the nature and volume of the data.

Rationale for automatic transfer of data is as follows:

- a. Delay times can be shortened.
- b. Reliability.
- c. Management of data (decreased complexity of the data accountability function).

3.2.6.2 Flight Dynamics. In general, the Flight Dynamics Computing functions are as follows:

- a. Attitude determination and control (if POCC not self-contained).
- b. Attitude and orbit maneuver prediction and command determination (station-keeping, achieving nominal orbit/deorbit, changing nominal orbit during mission (for example, AE)).
- c. Possibility of pointing user spacecraft to TDRS (if POCC not self-contained).
- d. Maneuver evaluation.
- e. ACS and sensor evaluation and parameter estimation (such as alignments and scale factors).
- f. Mission planning analysis.

For further definition of the Flight Dynamics Computing functions, Section 3 of Reference 1 should be consulted.

The POCC/Flight Dynamics Computing data interfaces are as follows:

- a. From POCC to Flight Dynamics Computing.
 - (1) Real-time telemetry parameters on a case-by-case basis.
 - (2) Attitude-related data.
 - (3) Mission and/or class-dependent data.
 - (4) Real-time Orbiter (and/or IUS/Tug) attitude and orbit ephemeris.
 - (5) Spacecraft health and safety parameters.
 - (6) Remote operations capability for Poccnet data bases.
 - (7) Report generation (maneuver requirements and desired activity timelines).
- b. From Flight Dynamics Computing to POCC.
 - (1) Real-time computed attitude results and information (such as gyro bias and spin vectors).
 - (2) Payload command information for implementing maneuver plans.

- (3) Flight-dynamics-related POCC OBC support information (such as star catalog updates).
- (4) Remote operations capability for Pocnet data bases.
- (5) Report generation (planned payload maneuver and activity sequence).

The mechanisms and requirements of the POCC-to-Flight-Dynamics-Computing transfer of real-time sensor telemetry data sets should be the same as those for the POCC-to-Orbit-Data-Operations transfer of stripped telemetry (refer to paragraph 3.2.6.1).

3.2.6.3 Mission Planning and Scheduling. Mission planning and scheduling aids (scheduling matrices, world maps, etc.) are now provided to the POCCs on a paper medium as are the scheduling aids generated by Command Memory Management. The POCCs have the following requirements:

- a. To access scheduling aids in the Mission Planning Data Base in much the same manner as accessing orbit ephemeris/vector data.
- b. A signaling capability if quick reaction is a requirement.
- c. For special scheduling aids generated on request.

3.2.7 DATA PROCESSING FACILITIES

Pocnet will interface with the following data processing facilities:

- a. Telemetry On-Line Processing System (Telops).
- b. Image Processing Facility (IPF).
- c. Spacelab Data Processing Facility (SDPF).

3.2.7.1 Telemetry On-Line Processing System. Telops is a NASA telemetry data collection and distributing system that is located at GSFC. Telops provides for initial capture of the telemetry data, for its recording in a form for subsequent use, for the decommutation of the processed telemetry into groupings relevant to a particular experiment, and for arranging for the transmittal of these data groupings to the experimenters by mailed tapes or possibly by communications lines.

The Telops input processor will be capable of supporting a peak 2.688-Mbps input rate (paragraph 4.16.1 of Reference 1). However, the total volume of processing

that is possible is 4.5×10^9 bits per day of telemetry data (paragraph 2.1.2 of Reference 1), which is an average of 52 kbps. The Telops mass storage system will hold 1.3×10^{12} bits of data (8×10^{11} telemetry bits).

Although presently in the formative stage, Telops envisions having an output processor system which will utilize communications lines for data transfer to the experimenters and will provide a conversational request mode to use when requesting data. If such a system is implemented, a Poccnet/Telops interactive interface will be easy to effect.

The rationale for Poccnet/Telops interactive communications is as follows:

- a. The POCC may require telemetry data to provide a quick-look capability with a maximum turnaround time of 24 hours (for example, Seasat). This can be most efficiently accomplished by direct data base access of the Telops data base.
- b. Telops may require sensor status and modes from housekeeping data for correlative and accountability purposes.
- c. POCCs may require access to archival data.
- d. The timeliness and reliability of transferred data (such as telemetry data needed by POCCs for quick look and sensor status and mode data required by Telops) will be enhanced relative to off-line (magnetic tape) methods.

3.2.7.2 Image Processing Facility. The major function of the IPF will be processing image data. At the present time, the data required by image processing at GSFC is tape recorded at STDN ground stations and then delivered to GSFC. If approved, a July 1978 acceptance test is proposed for Goldstone/GSFC and Fairbanks/GSFC Dom-sat hops for the transmission of Landsat data. (Presently, approximately 90 percent of the GSFC image processing effort is processing Landsat images.)

The only interface between IPF and the Telops will be that Telops will deliver tapes of image data to IPF as required to be processed. No other IPF/Telops interface is planned.

The current interface between image processing and POCC is that the former requires certain housekeeping data from the LandsatOCC as follows:

- a. Landsat has a separate attitude measurement system (independent of control) which is required to improve picture location accuracy. This attitude data

is stripped from housekeeping data and sent to image processing where attitude is computed. It is believed that in the future the MMS on-board navigation measurements will be sufficient for picture location accuracy.

- b. Image processing sensor status and mode, which are in housekeeping, are required for purposes of accountability. For example, if certain image data is not received, then image processing does not know a priori if there is an outage or if a sensor is off.

IPF feels that sensor status and mode will be required by IPF in the future. Whether this information is to be obtained by IPF via a data base access or by some other means such as a hand-delivered hard copy is TBD.

3.2.7.3 Spacelab Data Processing Facility. SDPF processing includes data reduction, formatting, and merging of the data with orbit and attitude information. The latter merging process may also require inputs from the mission command history and support computing systems. Processing of this type will be undertaken at SDPF from raw data received at GSFC directly from TDRSS. The additional information required for the merging process is available at the Spacelab POCC, although more refined attitude information may be required for some experiments. The Spacelab POCC-supplied data needs to be transferred to SDPF and this may be accomplished either by data transmission or delivery of data tapes, where the mode employed is dependent on the desired turnaround. If a rapid turnaround is required, the capability for transmitting this data to SDPF must be made available. For further information, refer to Reference 2.

3.2.8 DATA ACCOUNTABILITY SYSTEM

The Data Accountability System (DAS) is used to monitor data flow between STDN, the POCCs, and other users concerning all telemetry and tracking data movements and to prepare management information real-time displays and printed reports concerning data status.

The present DAS operations philosophy will undergo a change for the 1980s. At the present time, DAS fulfills its functions by accessing in real-time the flow of information through NASCOM to the POCCs and other users. In the 1980s, the DAS will collect information and compile a data base for evaluating the quality and quantity of payload support and to provide management information. Inputs to the DAS from the

POCCs and other users should be transmitted via intercomputer communications. These inputs may be in real-time and will consist of the following:

- a. Schedule information.
- b. Support analysis data derived from examining the telemetry stream.
- c. Other undefined data accountability reports.

3.2.9 MISSION OPERATIONS CENTER

The MOC Mission Control Room (MCR) displays mission-related information during launch and special maneuver phases of missions. The primary method of presenting the information required from a POCC to the MCR is via the CCTV system. Any normal television video output from the POCC can be patched and shown in the MCR. Thus, for example, the MCR can request a video display of the strip-chart recorders in a POCC. At the present time, the MCR communicates its requests to a POCC by voice link. There may eventually be a need for an interactive display or hard-copy terminal connections between the MCR and Poccnet in order to be able to access additional planning or mission data.

3.2.10 EXTERNAL EXPERIMENTERS

Interfaces will be required to support commanding and data handling functions for external experimenters on the range of payload programs to be supported by Poccnet. The experimenter facilities used in these support modes may be based at either GSFC or the home institution. These facilities and their required interfaces are additional to those supplied at the POCC, but the experimenter facilities may require an interface with the POCC facilities via Poccnet. This will be essential in the case of command functions, since all command operations will normally either be executed or verified via standard POCC procedures.

In the case of experimenter facilities based at the home institution, interfaces could be required to support either the transmission of command messages to the POCC or for interactive communications between the experimenter and a payload support computing system at GSFC (such as that used for the AE program). In this latter mode, the experimenter facility would be a remote terminal to this system and provide functions for updating data base information, program initiation, and control. The capability to output system information to the experimenter could be required to support

this mode. Other types of information possibly required by an experimenter at his facility include raw data dumps, instrument housekeeping and quick-look analysis outputs, relevant payload parameters, and orbit/attitude data. Standardized interfaces are therefore required to support this range of data types via Poccnet.

The development of GSFC-based facilities for quick-look scientific data interpretation and payload control is expected for a number of proposed payload programs, such as Atrex (see Reference 1). These quick-look facilities would be located separately from the POCC but would interface with the POCC via Poccnet in order to transfer payload commands or control requests. The generation of all payload commands would normally be undertaken at the POCC using standard procedures developed for the particular payload class concerned. GSFC-based facilities of this type would require access to the instrument data stream and to relevant payload, orbit, and attitude data generated or processed at the POCC. Communication lines from the quick-look analysis facility could be required to link one or more remote terminals located at the experimenter home institutions directly with the system. These communication lines would be used to transfer spacecraft, orbit, attitude, and housekeeping data and quick-look analysis outputs to the home-based experimenters.

3.2.11 MISCELLANEOUS

The Poccnet Inter-Process Communications system proposes to adopt the new national/international standard serial communication protocols. Thus, in effect, any POCC would be able to communicate with the world (that is, any Poccnet POCC can communicate with any demonstration or operational device which is on a network with protocol).

In the realm of public relations, Poccnet could easily provide terminals in the Visitor Center, real-time displays at the Sioux Falls Demonstration Center, special remote terminals during launch, etc.

Poccnet would allow GSFC to support remote integration and test facilities, lightweight portable POCCs, remote POCCs, Spacelab POCC Remote Information Centers, etc. The possibilities are endless.

3.3 MODEL POCC REQUIREMENTS

This section presents an estimate of the number of POCCs required at GSFC in the 1980s and their anticipated envelope of requirements.

3.3.1 NASA/GSFC MISSION MODEL

The mission model for this study is based upon the following references:

- a. Tentative OFT Mission Payloads List, May 5, 1976.
- b. Early STS Missions Plan - Cargo Manifest, April 1976.
- c. GSFC Mission Support List, April 1976.

The Payload Operations Control Center (POCC) Support of Shuttle Era Free-Flying Payloads Memorandum (Reference 1) attempts to accommodate the NASA free-flying payloads into the GSFC POCC facilities through CY83. The results of this study were tabulated by M&DOD and appear in Figure 3-12. From this figure, it is seen that a minimum of nine POCCs are required, where MSOCC is counted as one POCC. Actually, the MSOCC of the 1980s can be considered as four POCCs since it will be able to simultaneously support four payloads and 12 or more overall. Thus, a more precise method of counting POCC facilities is to count Mission Operations Rooms (MORs). On this basis GSFC must support up to 12 MORs by CY83 and perhaps twice as many through the 1980s. Note that Figure 3-12 does not consider the possible support of Spacelab missions.

The typical POCC must be able to simultaneously handle the receiving and processing of real-time telemetry, memory dumps or tape-recorder data dumps, and command generation and transmission. For an MMS-type spacecraft, the POCC must be capable of receiving, bit and frame synchronizing, decommutating, and processing a real-time telemetry stream at rates of up to 64 kbps. Even at this rate, however, the POCC would not be required to decommutate, convert to engineering units, limit check, display, etc., more than 8—10 kbps of housekeeping data in real-time. The rest of the data would be written to a storage medium for distribution and/or off-line playback. Memory dumps are received at the rate of 32 kbps and must be processed in real-time. Tape recorder dumps are received (for later processing) at the rate of up to 512 kbps. Memory dumps and tape recorder data dumps will not be received simultaneously (Reference 2). The POCC must be capable of generating and transmitting commands at the throughput rate of 2000 bps (about 42 commands per second).

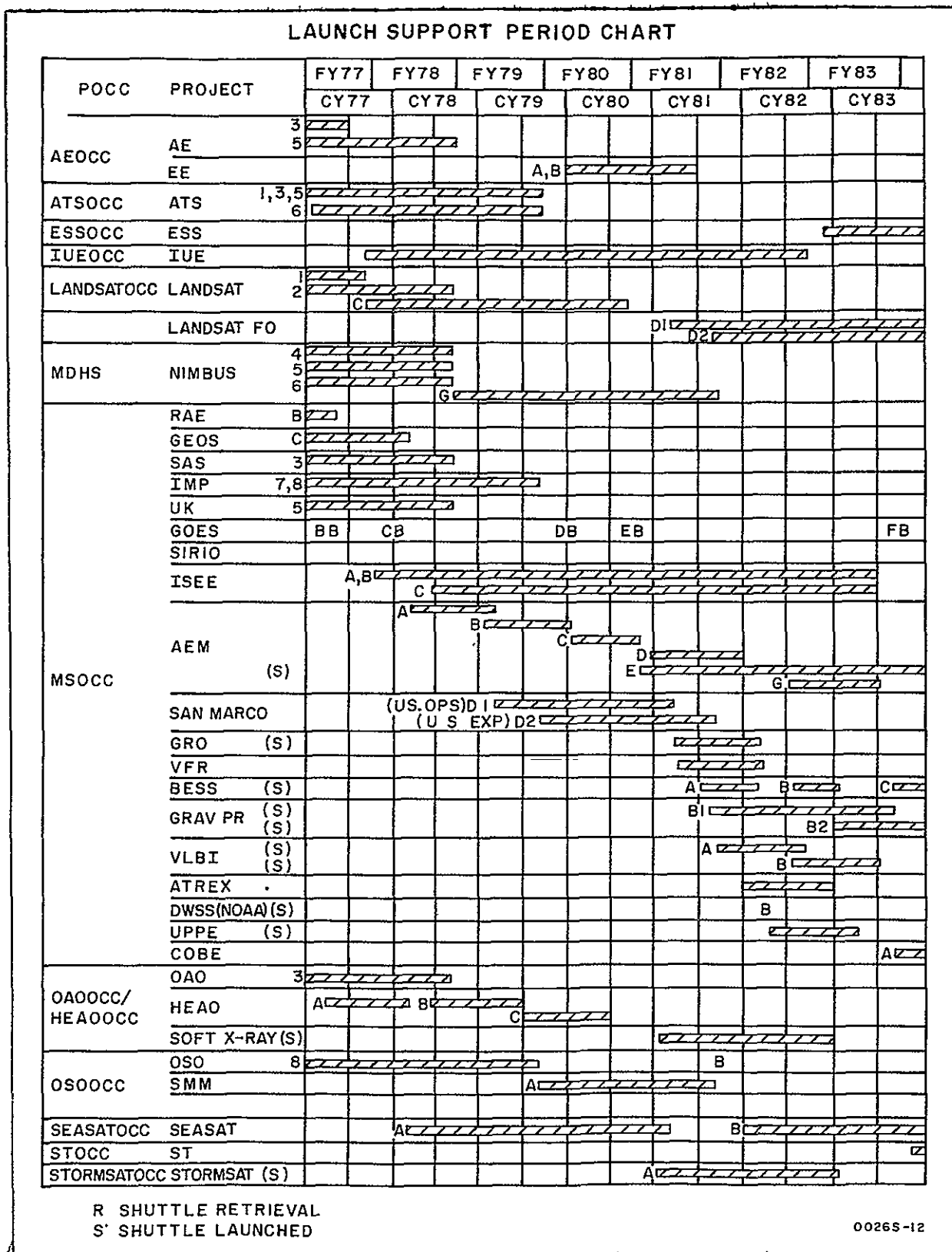


Figure 3-12. Launch Support Period Chart

3.3.2 MMS MODEL POCC

3.3.2.1 Introduction. The Model POCC is an abstraction or pure design of the capabilities all POCCs must have and how they should be implemented in the Poccnnet era. The purpose of the Model POCC study is four-fold. First, it serves to identify and describe POCC functional capabilities and requirements in the Poccnnet era beginning in 1980. Second, the Model POCC will act as a driver on Poccnnet design, thus identifying candidate software for modularization and standardization. Third, it is hoped that the Model POCC will demonstrate to the user community (projects) that Poccnnet is responsive to its mission requirements. Fourth, it is intended that the Model POCC will serve as a tool for familiarization with, and planning of, POCC designs.

The environment chosen for the Model POCC study centers on a low earth-orbiting Multimission Modular Spacecraft (MMS) as the payload and a Space Transportation System (STS) Orbiter as the launch vehicle. It is felt that the requirements generated by the MMS and STS are among the most complex as well as being highly representative of many payloads planned for the Poccnnet era. In addition, both the MMS and STS are reasonably well enough defined at this time so as to permit a meaningful study. Other factors such as TDRSS, JSC, KSC, and the various phases of a mission were analyzed in the course of this study.

3.3.2.2 Model POCC Functional Requirements. This section identifies and describes functions which the POCC must perform in order to properly support the payload and meet the requirements of the user community. These functional requirements were derived after analyzing the requirements of the MMS spacecraft, the STS, and NASCOM/STDN. In addition, representatives of the potential user community were contacted for comments and suggestions. The study drew heavily upon past and current control center experience with projects such as OAO, ATS, OSO, AE, Nimbus, and SAS.

The functional requirements were studied with respect to the mission phase. Requirements for integration and test, operations training and POCC validation, prelaunch checkout, and normal operations were identified and analyzed. Interestingly, this part of the study showed that the normal operations phase of a mission contained virtually all of the requirements identified in the other phases and some additional requirements.

All of the functional requirements listed are mission independent, thus making them good candidates for a standard, modularized software package. Many of the functions listed can and will be performed by the OBC. However, it is felt that any capability resident in the OBC should be available in the POCC in order to provide maximum flexibility in planning OBC utilization as well as coping with contingencies.

3.3.2.2.1 Subsystem Status Monitoring. The conversion and display of spacecraft telemetry in engineering units is required in both real-time as well as off-line (for example, to analyze history tapes or tape recorder dumps) in order to ensure spacecraft safety and to ascertain the health and status of the attitude control system (ACS), the command and data handling (C&DH) subsystem, the power subsystem, the propulsion subsystem, and the experiment subsystem.

Display formats should be innovative and have a high information density. The total number of display formats should be kept to a minimum by encouraging standardization. Display formats should be human engineered in order that data be "visible" and easy to read. Important information should be highlighted by different character fonts, the use of color, and different sizes. Printer formats should be very high density inasmuch as their prime use is for off-line analysis and recordkeeping (as opposed to real-time monitoring). Display formats should be easy to implement and modify in order to facilitate quick turnarounds. The ability to select different display formats rapidly in real-time is important.

3.3.2.2.2 Error Monitoring. Error monitoring of spacecraft telemetry and POCC system data includes the detection and display of both analog data which exceeds specified limits as well as discrete data which indicates improper status. Error displays should indicate time of occurrence, present value and/or status, and limit value. The idea of using audio alarms and annunciator panels should be explored.

3.3.2.2.3 Event Recording. Event recording includes the detection and display in chronological order of all changes in spacecraft status (discrete and pseudo-telemetry), all commands sent to the spacecraft, and ground system configuration instructions. In addition, the error monitor information should be included. The event record information should be displayed on a high-speed printer. The ideas of recording the event record data on a dedicated channel on the POCC history tape and incorporating a video display in the MOR for event data (with the ability to look back in time) should be considered.

3.3.2.2.4 Spacecraft Bookkeeping and Trend Data. There is a need in the following areas to record spacecraft telemetry, perform relatively simple calculations, and store and/or plot graphically the results:

- a. Momentum bookkeeping — Recording inertia wheel speeds as a function of time. Plotting axis vs. axis (e.g., roll vs. pitch). Used to determine optimum wheel unloading points, thereby reducing propellant consumption.
- b. Propellant bookkeeping — Recording tank pressures and temperatures periodically over a 24-hour period. Calculating propellant remaining. Recording jet pulses during various configurations to calculate and optimize utilization rates.
- c. Energy bookkeeping — Recording battery terminal voltages, temperatures, and charge/discharge rates with respect to time. Determining battery state of charge and predict time remaining to minimum permissible state of charge.
- d. Tape recorder bookkeeping — Keeping track of tape dumps (time, data, duration). Calculating time remaining.
- e. Thermal monitoring — Recording and plotting selected temperatures as a function of time.

3.3.2.2.5 Spacecraft Analysis Programs. There are several tasks to be performed which require complex calculations and handling of the spacecraft telemetry. These tasks require special applications programming support and, if not performed in, or controlled by, the POCC, require POCC direction and review to ensure correct results. A few examples are as follows:

- a. Attitude determination — This information may be required in the POCC in real-time to ensure that experiment objectives are met. In addition, operations personnel need to know spacecraft attitude with respect to the sun, earth, TDRS, etc.
- b. ACS sensor calibration and alignment — Telemetry from the IRU, star trackers, and experiment fine-error sensors (FES) needs to be recorded periodically and massaged to determined drift rates (and rate compensation data) and sensor misalignment data.

- c. Maneuver calculations — The POCC must have the ability to display in real-time, graphically and in printed format, spacecraft maneuver predicts, spacecraft telemetry and comparisons between observed and predicted data (this capability would be similar to the ATSOCC system for ATS-6).
- d. Orbit determination — Spacecraft orbit ephemeris data must be calculated periodically for updates to the OBC and the ground system. This may require the use of global positioning system (GPS) transponder data (assuming the payload has a GPS-compatible transponder).
- e. Power subsystem analysis — Special programs are required to monitor the performance of the batteries and solar array. These programs are considerably more complex than those that do simple bookkeeping. Long-term factors, such as number and depth of discharge cycles, are calculated and stored. Solar constant data, spacecraft attitude, and other factors are used in calculating and comparing current and predicted solar array performance.

3.3.2.2.6 OBC Image Store and Compare. The POCC will have the capability of commanding and processing total and partial OBC dumps and performing image compares. The POCC should have the capability of generating an OBC memory load to correct any image non-compares. This will require that the POCC maintain a current OBC image at all times.

3.3.2.2.7 Payload Commanding. Commanding of the payload will be performed with the Command Management System (CMS). CMS capabilities and requirements are described as follows:

- a. Structure — The CMS shall process commands via the following five processors:
 - (1) Command Memory Management (CMM) — CMM will process requests for stored commands from the experimenters (experiment mission planning) and the project (project planning/mission planning). CMM will create an OBC command memory load for uplink to the spacecraft.

- (2) OBC data block preparation — This program will prepare OBC programming information (instructions) and operational information for uplink to the payload.
 - (3) Manual commanding — This program will permit immediate creation and uplink of any command to the payload.
 - (4) Automatic sequence processor (ASP) — This program will enable mission operations personnel to create in real-time, near-real-time, or off-line a programmed set of instructions for use during real-time contacts with the payload. These instructions will be written in a standard GSFC STOL, developed during the Poccnet study (Appendix 3A). STOL instructions include spacecraft commands (real-time and/or stored), POCC computer instructions, and operator messages to the operations personnel.
 - (5) Real-time slew — This program will enable mission operations personnel to reorient the spacecraft attitude in real-time. The program will receive a pointing request, read the current spacecraft attitude from the real-time attitude determination program, prepare a set of real-time and/or stored commands, and uplink these commands upon request.
- b. Command types — The CMS will be capable of handling and creating real-time and stored commands. These commands may be either discretetes (such as equipment on/off) or data (such as OBC IRU drift compensation).
 - c. Interfaces — The CMS will receive instructions from experiment planning, mission planning, or mission operations in real-time or off-line via punched cards, magnetic tape, data link, CRT/keyboard, or command panel. Command timing must be controlled to the limits of each element within the network, the spacecraft, and the Poccnet.
 - d. Command checking and verification — Command checking involves examining all commands for violation of any operational constraints or restrictions as well as checking for accuracy. This will be achieved by driving a video display in the MOR and displaying all command formats and descriptors as well as violation flags.

The CMS must provide the means for command traceability by displaying milestone information in the POCC from key elements in the network (such as SMCC or ground station SCE).

Command verification is the examination of telemetry data for correct response as the result of executing a command. CMS output and verification messages will be displayed in the MOR on a video device and recorded by the event printer program (event recording). In addition, the CMS will provide an integrated timeline printout upon request showing the current status of the CMM and ASP.

The CMS will be capable of performing command memory load verification. This will be achieved by CMS commanding and OBC command memory dump and performing an image compare. All image compares and non-compare will be displayed (video and printer).

3.3.2.2.8 Data Base. The POCC should have the capability of reading, modifying, transmitting, and receiving data base information in real-time (that is, when the POCC system is up and running) as well as off line. The functional contents of the data base would include commands, telemetry, display formats, test and operations procedures, vehicle parametric descriptions, performance history, OBP data, operational checkpoint data, POCC statistics, and mission planning data. Configuration management of the POCC data base is a key POCC responsibility.

A Model POCC payload data base concept, to be used for MMS/SMM and the MSOCC 930 replacement systems, was developed as part of the Poccnet study (Appendix 3B).

3.3.2.2.9 Mission Planning. Mission planning is the merging of the experiment planning and project planning activities resulting in a valid and unified set of spacecraft commands and operations instructions which, when properly executed, will result in the achievement of experimenter objectives. It is expected that mission planning (which results in requests for network and Poccnet resources) will take place within the POCC. Mission planning personnel will require an operational system to perform the following:

- a. Access the experiment planning data base and store experiment planning requests (command sequences, special instructions, etc.).

- b. Access the project planning data base for payload operations ground rules, constraints, restrictions, procedures, etc.
- c. Access the POCC operational data base and other NASA center data bases (for example, SMCC).
- d. Build and store mission operations sequences (commands and instructions) and load the CMS.
- e. Check for violations of constraints and restrictions and read feedback from operations personnel.
- f. Transmit operations instructions (such as experimenter briefing messages) to experiment operations personnel via NASCOM.

Experiment planning may occur inside or outside of the POCC. Transmission of information between experiment planning and mission planning could be via data link or magnetic tape.

3.3.2.2.10 Network Scheduling. The POCC should have the capability of rapid and comprehensive communications with NASCOM/STDN. It is envisioned that the POCC should be tied in with the NCC scheduling system in real-time via a CRT/keyboard terminal and high-speed printer. Information to be transmitted to the POCC would be NASCOM/STDN facility status; confirmed POCC schedules for lines, stations, and station equipment; payload priority information; and network forecasts. POCC information to be transmitted to the network would include POCC schedule requests, POCC status, resolutions on schedule conflicts, requests for critical support, requests for support during a spacecraft emergency, ground system directives, and acknowledgments.

3.3.2.2.11 Poccnet Scheduling. The scheduling of Poccnet resources is a joint effort of the POCC and Poccnet Data Operations Control (DOC). Consequently, a POCC/DOC communication and scheduling system is required. The requirements would be similar to the network scheduling capability described in paragraph 3.2.2.12, that is, real-time operation and the use of a CRT/keyboard terminal and line printer for the transfer of resource schedule request and acknowledgment information.

3.3.2.2.12 Support Computing Facilities. For cost effectiveness, there may be tasks assigned to support computing facilities external to the POCC because of the infrequency with which they are performed, the size of the resources required for them, or because they clearly fall outside of the responsibility of the POCC. The POCC will require the means to communicate with these support facilities. The following are several candidate support computing facilities in the Pocnet era:

- a. Payload Simulator.
- b. Command Management.
- c. OBC Support Facility.
- d. Mission Planning.
- e. Data Management.

3.3.2.3 Model POCC System Control. The STOL is the medium by which the user communicates with and controls the POCC system. The STOL user functional requirements have been grouped into the seven areas described in the following paragraphs:

3.3.2.3.1 Telemetry Processing. STOL is required in the following areas to allow the user to control the processing of telemetry in the POCC:

- a. Deconvolve/decommutate (rate and format).
- b. Conversion (to engineering units).
- c. Generation of pseudo-telemetry.
- d. Computation of derived parameters.
- e. Error monitoring (analog limits and status).
- f. Control of test statements (<, >, =).

Most, if not all, of the above are configured at run time in today's systems. However, the ability of the user to monitor the status and to control each of these areas is necessary for proper control of the POCC system.

3.3.2.3.2 Display Control. STOL display control can be broken down into the following four areas:

- a. Display update rate.
- b. Routing (CRT, SCR, LP, hard-copy device, plasma panel, etc.).

- c. Display format.
- d. Type of data being handled (raw, processed, applications, or ground system status).

3.3.2.3.3 Information Storage and Retrieval. The POCC system must be able to control under STOL the storage and retrieval of the following kinds of information:

- a. Raw payload telemetry.
- b. Processed telemetry.
- c. Applications data.
- d. Experiment quick-look data.
- e. Ground system data (status and inputs).
- f. Data contained in the Data Base Storage System.

3.3.2.3.4 Command Management System. The command management functions under STOL fall under the following six categories:

- a. Manual, real-time payload commands.
- b. Command Memory Management — Includes the interface between experiment and mission planning and the CMS as well as the fabrication and uplinking for payload commands.
- c. Automatic sequence processor (ASP) — This function was previously known as autopilot. ASP provides the capability of creating processors and schedules in order to automate payload commanding and controlling POCC system configuration and provides a means of communication between the system and the user.
- d. OBC data block preparation — This part of the CMS enables POCC operations personnel to create and uplink information to the OBC such as ephemeris data and sensor misalignment data as well as programming instructions.
- e. Real-time slew and antenna pointing — By using either real-time telemetry or manually input data, the CMS will calculate and command real-time slews or antenna commands to the payload.

- f. Command checking and verification — The CMS will check all command requests for constraint violations and notify the user if a command has been previously designated as restricted or critical prior to uplinking. In addition, the CMS will recognize and display command recognition patterns (milestones) and perform command verification using payload telemetry.

3.3.2.3.5 POCC Data Processing. POCC data processing encompasses most of the higher order processing of telemetry and applications programming. The STOL will enable the user to control and interpret this area of POCC software which includes the following:

- a. Attitude determination.
- b. Maneuver calculations.
- c. Sensor calibration/alignment.
- d. Orbit determination.
- e. Power subsystem analysis.
- f. Bookkeeping (momentum, propellant, energy, tape recorder, etc.).
- g. Trend data (thermal, pressure, voltage, etc.).
- h. Experiment data processing (quick-look and low data rate instruments).
- i. Payload unique processing.
- j. Scratch pad programs/processors.

3.3.2.3.6 POCC External Communications. The STOL must also be capable of enabling the POCC user to communicate with other systems and users who interface with Poccnet and the POCC. This world outside of the POCC includes the following:

- a. DOC.
- b. NASCOM.
- c. NCC (STDN).
- d. Telops.
- e. DAS.
- f. Operational support computing.

- g. DBS.
- h. OBC Support Facility.
- i. Payload simulator.
- j. Support Computing Facility (DEMOS and 360s).
- k. External experimenters.
- l. Other NASA centers.

3.3.2.3.7 POCC System Control. In addition to the user functional requirements, the STOL must also enable the POCC user to control the basic elements of the POCC operating system. Examples in this area would include system initialization, control of peripherals, and job priorities.

3.3.2.4 Model POCC System Improvements This section identifies and briefly describes five areas of POCC system capabilities where present control center systems are weak. It is felt that if significant progress were to be made in these areas, then future POCCs would be much more efficient (in terms of minimizing wasted man-hours, ground systems time, and payload operating time) than present ground systems.

3.3.2.4.1 Multiple-Terminal Access. The POCC system should be capable of communicating or responding to more than one STOL user terminal (such as the CRT/keyboard) at a given time. Present systems are highly restrictive in that they will only allow one terminal at a given time to have run-time control. This results in the POCC users waiting for each other and doing most tasks serially. A system allowing multiple-terminal access to do things (such as STOL initiating procedures) simultaneously would considerably speed up POCC data processing. It is recognized that such a system will lead to more complexity in software design, but the concept is feasible. Techniques such as spooling will have to be employed to handle competition for limited resources such as line printers.

3.3.2.4.2 Multiple-Processor Capability Within Each Terminal. The concept of multiple-processor capability within each terminal is merely the application of the idea expressed for multiple-terminal access as applied to an individual terminal. The limitation that an individual user at a given terminal can only request or direct one

processor at a time is highly restrictive. The user should be able to assign a task to the POCC system and, while the system is performing this task, be able to communicate with the system in order to assign other tasks or make requests. For example, the ground system should be able to uplink a lengthy command memory load and simultaneously take a full set of subsystem snapshots.

3.3.2.4.3 Ground System Status Monitor. The POCC system should have displays in the MOR for monitoring POCC hardware and software status. Most of today's ground systems are poor with respect to internal system visibility. As a result, the user must always remember what he has asked the system to do and use that information to deduce the system configuration. This leads to much confusion, especially when part of the system hangs up or if someone else changes the system configuration. In addition, if the concepts of multiple-terminal access and multiple-processor capability are employed, the ground system monitor takes on added importance in helping the user to supervise POCC system status and workload.

3.3.2.4.4 Scratch Pad Core and Storage. Provision should be made in the POCC system for special-use programs to support mission operations and payload analysis. These programs will be mission unique and will fall outside of the tasks identified in paragraph 3.3.2.2. An example of such a program might be one to cope with a payload subsystem failure. Such a program might require real-time telemetry and would perform calculations and automatic real-time commanding under POCC supervision. Programs such as these could easily be implemented if scratch pad core and storage (such as disk) were available. It is expected that computer system hardware costs in the Pocnet era will be relatively inexpensive. Thus, the scratch pad core and storage area should be modularized and made to be easily expandable as needs require.

3.3.2.4.5 Human-to-Machine Interfaces. The human-to-machine interfaces in the POCC occur principally between the user and the I/O devices. Output devices in today's POCCs consist mostly of CRTs, line printers, hard-copy devices, and wall-board displays. Input devices are mainly keyboards, teletypes, card readers, paper tape readers, magnetic tape transports, and pushbutton panels. It is interesting to observe that tasks which take place solely within the machine occur with unbelievable swiftness. However, when the machine must communicate with the user or vice versa, the transfer of information across this boundary moves at a snail's pace. Therefore, it would seem that the human-to-machine interfaces in the POCC are fertile areas for improving POCC efficiency.

The tasks which POCC users (that is, mission operations personnel) must perform in the POCC consist chiefly of information gathering (requests from experimenters, etc.), comprehension, decisionmaking, and data transfer (I/O) between the user and the POCC system. These are the four areas where the POCC system development effort should be concentrated to improve POCC system effectiveness and efficiency. Such a development effort should be supported by a prototyping program where new hardware and implementation concepts could be tried out and proven on a small scale before a system-wide commitment is made. In addition, early Pocccnet system design must not be restrictive so that future technological developments can be easily implemented in the system. It is expected that many such developments will occur over the next 15 years and Pocccnet should allow continual evolution and improvement.

Some examples of current concepts and technology which should be evaluated for Pocccnet in order to improve the human-to-machine interface are as follows:

- a. Plasma panels for displays and information output.
- b. Voice data I/O terminals (voice recognition and voice synthesis).
- c. Color CRTs.
- d. Annunciator panels.
- e. Distributed data base mass storage devices.
- f. POCC located command memory management system.

3.4 SECTION 3 REFERENCES

Paragraph 3.2.1

1. GSFC, Payload Operations Concept and Requirements for Earth-Orbiting Automated Payloads Launched by Space Transportation System, May 1976.
2. GSFC, JSC/GSFC POCC Command Interface Control Document, December 1976 iteration, Attn: J. Michael/GSFC.
3. J. Michael/GSFC, conversations.
4. GSFC, the second quarterly report of: Concepts Study of the Mission Operations Requirements for STS/Payload Synchronous Orbit Missions. The entire document will appear in February 1977.

Paragraph 3.2.2

1. GSFC, Spacecraft Tracking and Data Network Technical Manual, DDPS Phase II, Interface Control Document, MM-4387, Revision 3, August 1976.
2. GSFC, Networks 1980's Concepts Tutorial for Poccnet Study, October 1976. Speakers: W. B. Dickinson/840, E. Ferrick/860, C. D. Roberts/850, R. E. Spearing/805, and H. G. Theiss/810.
3. GSFC, Performance Specification for Telecommunications Services via the Tracking and Data Relay Satellite System, S-805-1, November 1976.
4. GSFC, STDN Operations Concepts 1980-1990, STDN No. 106, August 1976.
5. GSFC, Tracking and Data Relay Satellite System (TDRSS) User's Guide. STDN No. 101.2, Revision 2, May 1975.
6. GSFC, Command Operations through TDRSS Memorandum, M. E. Shawe/511, April 29, 1976.
7. GSFC, Mission and Data Operations Concepts, R. Adams/512.

Paragraph 3.2.3

1. GSFC, Payload Operations Concept and Requirements for Earth-Orbiting Automated Payloads launched by Space Transportation System, May 1976.

Paragraph 3.2.4

1. GSFC, Payload Operations Concept and Requirements for Earth-Orbiting Automated Payloads Launched by Space Transportation System, May 1976.

Paragraph 3.2.5

1. OAO Corp., Astronomy Spacelab Payloads (ASP) Operations Study, Phase II, Operational Requirements for the ASP Missions, September 1976.
2. OAO Corp., A Preliminary Assessment of the POCC Requirements for Supporting the ASP and AMPS Missions, November 1976.

Paragraph 3.2.6

1. Computer Sciences Corporation, CSC/TM-76/6105, Flight Dynamics Resource Requirements for the TDRSS/STS - Shuttle-Era Payload Support, ATR: D. Ketterer, Code 581.4, May 1976.

Paragraph 3.2.7

1. GSFC, Telemetry On-line Processing System (Telops) Interface Control Document, September 1976.
2. OAO Corp., Astronomy Spacelab Payloads (ASP) Operations Study, Phase II, Operational Requirements for the ASP Missions, September 1976.

Paragraph 3.2.8

1. Paul Myers/510, Conversations, December 1976.

Paragraph 3.2.9

1. R. H. Adams/510, Conversations, December 1976.

Paragraph 3.2.10

1. OAO Corp., Feasibility Study for Near-Real-Time Data Analysis Facility for the Proposed Temporal Explorer Mission, December 1976.

Paragraph 3.3.1

1. GSFC, Payload Operations Control Center (POCC) Support of Shuttle Era Free-Flyer Payloads, Memorandum from D. L. Fahnestock/513, June 7, 1976.
2. GSFC, Low Cost Modular Spacecraft, X-700-75-140, May 1975.

Paragraph 3.3.3

1. OAO Corp., Astronomy Spacelab Payloads (AS) Operations Study, Phase II, Operational Requirements for the ASP Missions, September 30, 1976.
2. Memorandum from MSFC/JA01/O. C. Jean to JSC/PA/H. E. Gartrell, "Spacelab POCC Requirements," April 14, 1977.

SECTION 4
SYSTEMS ENGINEERING

SECTION 4 SYSTEMS ENGINEERING

CONTENTS

<u>Paragraph</u>		<u>Page</u>
4.1	General	4-1
4.2	Systems Engineering Analyses	4-1
4.2.1	Systems Drivers	4-2
4.2.1.1	Driver 1: Standard Spacecraft, Shuttle, and TDRSS Orientation	4-3
4.2.1.2	Driver 2: Flexibility	4-3
4.2.1.3	Driver 3: Low Cost of Each POCC	4-5
4.2.1.4	Driver 4: POCC Autonomy	4-5
4.2.1.5	Driver 5: Orientation to People	4-5
4.2.1.6	Driver 6: Advances in Computer Technology	4-6
4.2.1.7	Driver 7: Evolutionary Organization and Operation	4-9
4.2.2	Systems Concept	4-12
4.2.2.1	Analysis of External Requirements	4-12
4.2.2.2	Analysis of Systems Drivers	4-14
4.2.2.3	Synthesis of Systems Concept	4-15
4.2.2.4	Summary	4-20
4.2.3	System Requirements Analysis	4-20
4.3	Systems Studies	4-22
4.3.1	IPC Simulation	4-22
4.3.1.1	Summary of IPC Simulation Results	4-22
4.3.1.2	Further IPC Study Activity	4-23
4.3.2	Systems Implementation Language Study	4-23
4.3.2.1	Summary of Consultant's Recommendations	4-25
4.3.2.2	Study Manager's Recommendations	4-25
4.4	Software Engineering	4-27
4.4.1	Software Engineering Standards	4-28
4.4.2	Common Software Steering Group	4-34
4.4.2.1	Software Engineering Standards Subgroup	4-35
4.4.2.2	Mathematical Analysis and Software Subgroup	4-36
4.4.2.3	Ground Systems Software Subgroup	4-36

ILLUSTRATIONS

<u>Figure</u>		<u>Page</u>
4-1	Systems Engineering Approach Used in Poccnnet Study .	4-2
4-2	Endless POCC Systems Evolution	4-12
4-3	Systems Concept Synthesis	4-13
4-4	Summary of the Application of Systems Drivers to the Synthesis of a Systems Concept	4-16
4-5	Autonomous Model POCC	4-20
4-6	Fully Interconnected Model POCC	4-20
4-7	Computerworld Article on Government Acceptance of DOD-1	4-26
4-8	Recommended Composition of Software Engineering Panel	4-30
4-9	Standard Approach Overview	4-31

SECTION 4

SYSTEMS ENGINEERING

4.1 GENERAL

The purpose of systems engineering is to create a total systems concept which best satisfies expressed requirements. The scope of systems engineering extends to subdividing the total system into identifiable, independent subsystems. Each subsystem must satisfy its own set of requirements derived from the system requirements as part of the systems engineering analysis.

Systems engineering is thus responsible for developing subsystem requirements, for defining and maintaining the interfaces between subsystems, and for overall systems assurance. Each of the subsystems of the overall GSFC POCC system concept is itself a "system" and this term is used throughout this report in preference to "subsystem".

4.2 SYSTEMS ENGINEERING ANALYSES

This section presents the systems engineering analyses and rationale for subdividing the overall GSFC POCC system requirements into requirements on the systems within Poccnet.

The systems engineering approach followed in the Poccnet study is shown in Figure 4-1. The external requirements are identified and described in functional and quantitative terms as appropriate. These requirements were presented in Section 3.

Next, generic systems drivers are applied. These drivers are generalized objectives or constraints which must be satisfied by any system chosen to meet the requirements. Unlike the external requirements, the drivers are chosen by system technical and management personnel to represent the desired direction of systems development. The systems drivers chosen for the Poccnet study are described in paragraph 4.2.1.

The drivers narrow the field of all possible systems concepts down to a small set of candidates capable of satisfying the requirements and responding to the drivers. This analysis of systems concepts is described in paragraph 4.2.2.

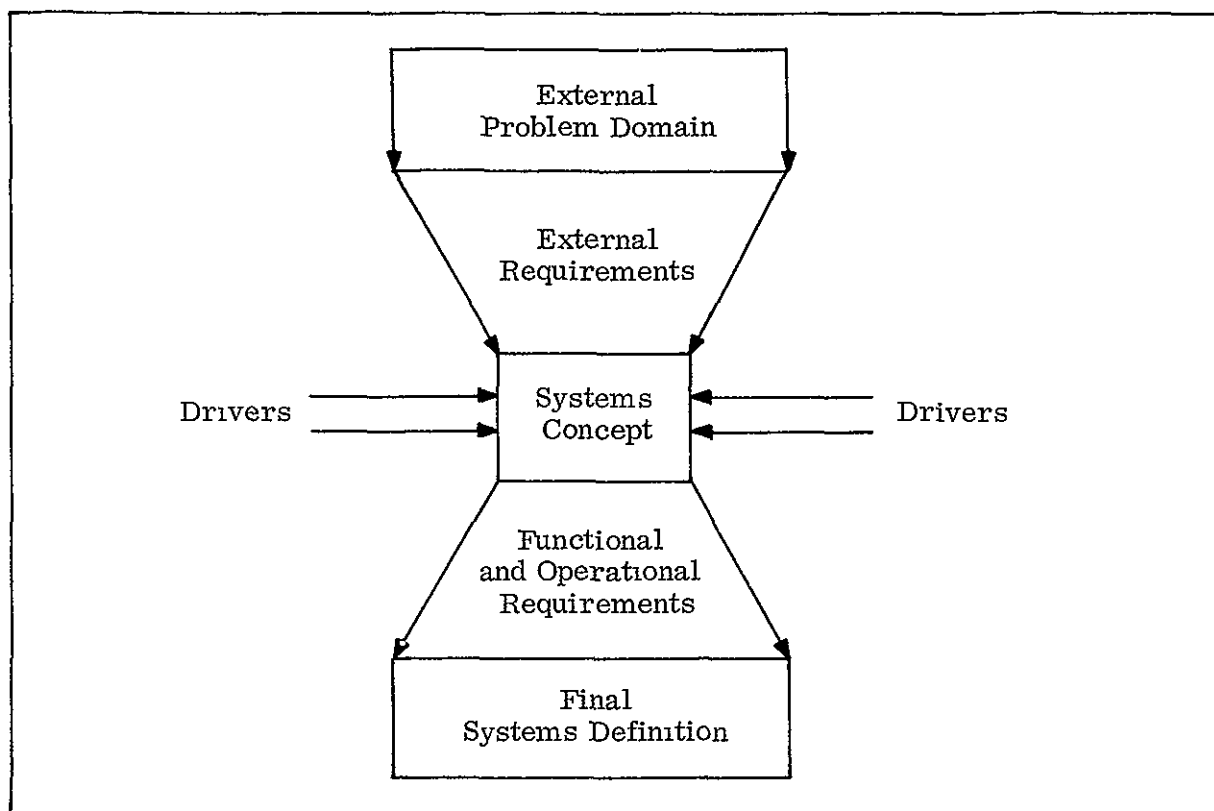


Figure 4-1. Systems Engineering Approach Used in Pocnet Study

The external requirements are then analyzed and applied to the systems concept candidates. This process generates functional and operational requirements on subsystems of the concept. It also stimulates specific engineering tradeoff analyses which must be performed to finalize the systems definition. This systems requirements analysis is presented in paragraph 4.2.3.

4.2.1 SYSTEMS DRIVERS

In the Pocnet Concept X-document, five drivers were listed as fundamental. These drivers are repeated here and described in full because of their importance to the study.

Two additional drivers, advances in computer technology and evolutionary organization and operation, were also recognized during the systems definition phase. These new drivers were implicit in the conceptual phase and are simply being rendered explicit herein.

The driver, advances in computer technology, ensures the economic and technical feasibility of any systems concept resulting from the study and declares that recent advances in technology must be brought to bear on GSFC POCC systems of the future.

The driver, evolutionary organization and operation, ensures that 15 years of mission support experience at GSFC are factored into the study and that the lessons learned in building and operating GSFC POCCs form the basis of the POCC systems concept of the future.

4.2.1.1 Driver 1: Standard Spacecraft, Shuttle, and TDRSS Orientation. Much of the justification claimed for the new generation of flight and ground systems is that they will provide better service to the user at lower cost than their predecessors. For example, the Multimission Modular Spacecraft (MMS) is a system consisting of a modular spacecraft standard bus and subsystems packages which can be easily and economically customized by a user project to support virtually any Shuttle- or Delta-launched free-flyer mission. A small standard spacecraft capable of being launched by the Scout vehicle is also being developed at GSFC under the Applications Explorer Missions (AEM) Project.

However, the benefits claimed for these standard systems are not obtained automatically. The user must be properly organized to take advantage of the new environment. The Shuttle data system, the TDRSS bent pipe, and the MMS 72-hour autonomy all require new ground support concepts and systems. The users must expect costs to occur at the initial stage, with the justification being that the new flight and ground systems provide a standard envelope of requirements and a standard interface. If standard POCC systems and applications are properly designed and implemented initially, they can be reused with, at most, minor modification for several missions. Thus, the cost is amortized over a number of users instead of being borne fully by each user. Therefore, Poccnnet is driven by standard spacecraft, Shuttle, and TDRSS orientation.

4.2.1.2 Driver 2: Flexibility. Most computing machinery has a useful life in the 10- to 20-year range. But most missions last only 0 to 5 years. Thus, POCCs are forced by simple economics to reuse equipment. The software sometimes costs even more than the hardware, and a substantial amount has approximately the same useful life (most software does not have an infinite life due to the life cycle of the application as well as the evolution of software technology). Therefore, from mission to mission, POCCs are also forced by economics to reuse as much software as they can. For these reasons, designing long-term flexibility into the POCC software and systems initially so that these can be easily and inexpensively reconfigured from mission to mission is a Poccnnet driver.

Equally important from the project's point of view is short-term flexibility during the mission. POCC resource requirements during the launch and early mission phases are usually much greater than during normal operations. Also, when contingencies arise or special operations are planned, it may not be possible to put sufficient resources together quickly to handle the situation properly. Traditionally, the only practical alternative to flexibility for most POCCs has been to plan for the worst case, that is, the case requiring maximum resources. This is a very costly approach, since much of the time the equipment is under-utilized. Obviously, a system which enables each POCC to quickly and easily reconfigure its systems to meet new or changing requirements while allowing another user to share equipment not needed for normal operations would save money.

Flexibility is required also because of the uncertain mission load in the future. Since it is difficult to predict the number of missions to be supported more than 2 years into the future, it is important that POCC systems be incrementally expandable so that fluctuations in support requirements can be accommodated. Such flexibility must be manufacturer and technology independent to be able to take advantage of price competition and technological advances when systems expansion is required.

Still another reason for flexibility is to allow quick response to changing requirements at minimum cost. This quick responsiveness type of flexibility is important, for example, for the support of shuttle-launched payloads which will have shorter lead times than most payloads supported in the past. Finally, flexibility is required so that systems may be upgraded to benefit from future advances in technology with minimal impact on running systems.

To summarize, flexibility is necessary on two counts. First, flexibility within the entire ground support complex is required in order to be able to adapt quickly to uncertain future mission requirements and systems technologies without undue cost or severe disruption to existing systems. Second, flexibility is required within the individual POCC computational system to economically provide the resources required to support various mission phases and to handle emergencies.

4.2.1.3 Driver 3: Low Cost of Each POCC. POCCs have traditionally cost only a few percent of the cost of the spacecraft series being controlled. In the 1980s, as the cost of the flight systems decreases and the complexity of their requirements increases, the POCCs will be under severe cost pressure. GSFC will be expected to take full advantage of opportunities for reducing cost arising from the standardization of the flight and ground systems with which the POCCs support or interface. Furthermore, the NASA budget is likely to remain tight in the foreseeable future. Therefore, Poccnet is driven by the low cost of each POCC.

4.2.1.4 Driver 4: POCC Autonomy. The usual systems response to drivers of flexibility and low cost is to pool resources and draw on them according to need. A method of resource allocation strategy is implemented to decide priorities and allocations. A very powerful countervailing driver to this traditional approach is POCC autonomy. A POCC has the complete responsibility for the spacecraft health and safety. The project should not have to jeopardize that spacecraft for arbitrary reasons based on the constraints of pooled operation. The project should always be able to operate its POCC as it deems necessary for legitimate spacecraft health and safety reasons, without being unduly concerned about other POCC operations running simultaneously. Any POCC system which cannot assure a project that it will have control of the resources it needs for survival of its spacecraft is not a satisfactory system for conducting space operations. Therefore, Poccnet is driven by POCC autonomy.

4.2.1.5 Driver 5: Orientation to People. The direct personnel costs of implementing and operating a POCC for an extended period of time far exceed the equipment costs. The maximum impact that can be made on costs is in the personnel area. Implementation personnel for software, hardware, and spacecraft systems today are sometimes forced to expend time and money just to work around system inadequacies. Systems difficult to program, integrate, operate, and maintain require expert attention. Personnel are frequently required to spend an undue amount of time in a nonproductive capacity, waiting for a printout, figuring out a cabling diagram, diagnosing an anomaly without proper instrumentation, etc. Yet studies have shown that in software engineering, productivity (production per person) varies inversely with the size of the project workforce, due principally to the increasing

overhead required simply to communicate information concerning the system under development. A large cost saving can be achieved by a higher efficiency software implementation using less manpower.

Operations personnel for both project and systems operations may also spend much of their time making up for deficiencies in the system. Some personnel are required to routinely handle many reels of computer tape a day in order to move a small amount of data from one computer to another. Other personnel must punch cards, run an entire data base generation program, and obtain a complete computer listing to make changes to a few items in a data base. Engineers on the consoles in the Mission Operations Room may have to work through an operation manually, even though it is simply an exact repetition of something they have accomplished correctly many times in the past. Inadequate instrumentation makes systems troubleshooting an art requiring well trained and highly paid experts.

Attacking personnel costs head on means that systems have to be oriented not only to technical requirements but also to the requirements of the people who have to develop, use, and maintain them. A system has to provide the man-machine interfaces and capabilities each user requires to be productive on the system. Therefore, Poccnet is driven by orientation to people.

4.2.1.6 Driver 6: Advances in Computer Technology. Two phenomena of immense importance to computer systems engineering are manifest at this point in time. They are, first, the rapidly declining cost of digital logic elements and systems and, second, the rapidly declining cost of computer interconnection via high-performance bit serial mechanisms.

When digital logic was expensive to build, the entire system catered to it. A computer central processor unit (CPU) was a very expensive resource, so a system could not afford to have very many of them. As a result, complex systems engineering designs were required to solve system problems. The problem had to be routed by wires to the CPU, where the problem was solved, and the solution was sent back to where the problem had arisen. Computer memory was also expensive when magnetic cores had to be strung on wires by hand. Hence, complicated mechanisms for cutting and folding problems had to be devised to fit them within available memory.

Now the cost of CPUs and memory are rapidly decreasing because of the plummeting cost of digital logic elements. This allows a vendor to sell many CPUs of the same basic design and use semiconductor memory in preference to core. The principal expense in systems development is now getting to be the cost of paying expert people for systems engineering design and implementation.

Therefore, rather than bringing all the problems into one central CPU for solution, it is beginning to make more economic sense in many cases to buy additional CPUs and place them where the problems are. For example, the bit-level and micro-second time-critical nuisances of handling devices may be left to minicomputers located near the devices, rather than tying up a larger, more expensive system. Microprocessors can be built into systems to do specialized jobs inexpensively, such as making a command pushbutton panel generate text strings to look like a keyboard.

With memory prices falling, the systems engineer can afford to buy enough memory to solve problems easily, without a lot of costly overlaying and data management.

The end result of falling logic prices is that it has become economically feasible and desirable to distribute a number of computers to solve complex distributed problems. By solving each subproblem where and as it arises, the overall system problem never becomes overwhelming or unmanageable.

The second key ingredient to make distributed systems technically and economically feasible is the ability to interconnect the distributed computers easily and at low cost. This is where the new high-performance bit-serial synchronous communications technology is so important.

The fundamental desirability of serial communication between computers in preference to parallel should be clearly understood. In existing GFSC POCCs, one finds computers with word lengths of 16 bits, 24 bits, and 32 bits, and character sizes of 6 bits and 8 bits. Selecting a parallel channel width for intercomputer communication poses a dilemma. Using serial channels seems a simple answer to the problem, provided the transfer bit rate is high enough to satisfy throughput and response requirements.

GSFC experience with the SCADI system, which hosts a variety of serial channels at rates up to several million bits per second, has shown the desirability of serial channels. Serial channels can be made to operate reliably over long distances

simply by adding line drivers. Extremely long distances of hundreds or even thousands of miles can be accomplished by lowering the bit rate to 9600 bits per second and using modems to relay the information through common carrier lines.

Serial channels seem to carry a bonus in reliability as well. Serial data arrives at its destination through essentially two simple wires, a data wire and a clock wire. Time lag variations across a bundle of parallel wires because of differences in line terminations and driver amplifier characteristics are no problem.

Most important of all is the fact that the data exchange is explicit and sequential. The sender must consciously select the data and transmit it; and the receiver must consciously receive the data and dispense with it. There is no opportunity for one computer to write in another's domain without its cooperation. The computers can arbitrarily scrutinize the communicated data and thereby defend themselves against either malfunction or malice. No external real-time interrupts arrive with, and in parallel to, the data, exposing an irreproducible software error. The sequential nature of the serial information transfer avoids subtle hardware and software problems in much the same way that avoiding go-to statements in structured programming improves software reliability.

Hardware serial channels can readily be built to work at 6 Mbps. These hardware channels can be full duplex and can include all logic necessary to make a direct memory transfer to or from the attached computer. Error checking can also be handled by the channel. Coaxial cables can handle these rates directly without modems at distances up to one kilometer.

A low-cost intelligent channel can also be built around a microprocessor. The microprocessor can take care of flow control as well as the other functions handled by the hardware channel. The microprocessor channel also has the advantage of allowing changes and adjustments to the communications protocol without changing the hardware. A microprocessor channel capable of bit rates of 2 Mbps or greater full duplex can be obtained commercially for under \$2000.

Given the advantages of serial interconnection, computer manufacturers and users have developed a number of high-performance protocols for transferring information between computers. Several of these protocols are in the final stages of standardization activities, and integrated circuit manufacturers are already producing microprocessors to implement these protocols. The \$1000 serial channel running at 1 Mbps will be available commercially long before 1980.

All the computer vendors will offer this standard interconnection technology as part of their product line, and their operating systems will support distributed computation structures to solve distributed problems as a matter of course. This technology, together with the falling cost of digital logic, will have an enormous impact on systems engineering. Therefore, Poccnet is driven by advances in computer technology.

4.2.1.7 Driver 7: Evolutionary Organization and Operation. GSFC has over 15 years of experience in designing, building, and operating POCC. Many of the existing GSFC POCCs had their genesis in the early 1960s. In those days, each new series of spacecraft had radically different characteristics and requirements. Also, the space business was in its fledgling years, and there were honest differences in philosophy as to how a spacecraft was best operated. As a result, each new spacecraft series led to a new POCC development. The Nimbus, the Orbiting Astronomical Observatory (OAO), and the Orbiting Solar Observatory (OSO) required that their POCCs be developed to meet the exact requirements of the spacecraft which they operated. As subsequent spacecraft in each series were built, they were operated from the POCC which had supported the earlier spacecraft of that series. Hence, the GSFC POCCs have been fully utilized over the years, and each POCC has responded efficiently to the requirements of its spacecraft series.

In addition to the POCCs dedicated to each spacecraft series, there evolved one POCC, the Multi-Satellite Operations Control Center (MSOCC), to service all those spacecraft which did not require a dedicated POCC because of the nature of their support requirements. Some of these were spacecraft with low commanding activity, so that support consisted essentially of taking data periodically to verify satisfactory spacecraft operation. Others were spacecraft with short-term support requirements, such as spacecraft for which NASA provided launch and initial operations services before turning normal operations over to another operating agency or country. MSOCC has been cost-effective for this class of requirements because it has maximized the sharing of resources and the reuse of equipment.

In fact, the MSOCC role has continually evolved to the point where MSOCC is currently supporting up to a dozen operational spacecraft, together with providing launch vehicle support for all the Delta launches since SMS-1. This level of support has strained MSOCC resources far beyond the most optimistic early expectations, yet MSOCC has never failed to provide support for an assigned mission.

Although the existing dedicated POCCs and the MSOCC have been successful, further evolution of POCC organization at GSFC is now required. This is because a number of POCC systems drivers are presently converging on GSFC for which the existing systems organization is unable to respond effectively. These systems drivers include the following:

- a. The equipment in several of the existing POCCs is outdated, decreasingly reliable, inadequate for upcoming missions, and no longer cost-effective. It will require reburbishment or replacement in the next few years.
- b. The flight and ground systems with which the POCC must interface are becoming more standard, yet more complex. With the standardization appears a new, seemingly paradoxical requirement: quick response to unknown demands on the payload operations system. As the payloads get easier to implement, the pressure will build to make the operations system more flexible and responsive.
- c. The nature of computing systems is changing. The hardware (which depends on technological costs) is becoming much less expensive, while applications software and operations (which presently depend on manpower costs) are becoming much more expensive.
- d. Cost is a major criterion in POCC systems development because of the continuing pressure on total NASA funding.

The nature of evolution as a system characteristic must be well understood for its importance to GSFC POCC systems to be grasped. Evolution is the growth and change of systems by an endless succession of small perturbations rather than by a few large discontinuities. Any given perturbation changes the system only a little, yet the cumulative effect of many changes is that a distinctly different system comes into being over a long period of time.

The GSFC POCC system of the future must be evolutionary in three distinct, important ways.

First, the system must represent an evolutionary growth from existing systems. The GSFC approach to POCCs has worked well in the past and therefore must not be abruptly abandoned. The strengths of the existing systems, principally POCC

autonomy and responsiveness to project requirements, must be retained. As new POCCs are required, however, small changes from existing systems approaches must be made to increase flexibility, decrease cost, and incorporate standard solutions to common requirements.

Second, any new systems which are required must be designed to allow evolutionary growth. This characteristic of systems may be called "modular responsiveness to unknown requirements". A system designed for evolutionary growth would allow modest new requirements to be met by small increments in system capability, while more difficult new requirements might create larger perturbations. However, if the system is properly designed to allow evolutionary adaptation, it should never be the case that small changes in requirements cause major system perturbations. (This driver is the death knell to large, centralized, integrated POCC systems.)

Third, the total GSFC POCC system must allow evolutionary operation. Too often in the past, systems operation has received little or no attention from systems architects. Yet many of the best recent systems have been designed by writing the user manual first, in effect, establishing operational user specifications as the principal system specifications.

Future GSFC POCC systems must extend this trend to the point of including evolutionary systems operation as a system requirement. This means that systems must be designed to be operated in a simple, straightforward way as the basic operational mode. Then, in addition, the system must be designed to allow the operations personnel to improve systems operation by evolving extensions to the basic operational modes. These extensions would come about through operational experience rather than being simply the ideas of systems designers who do not really know the operational problems.

This approach to operations design of systems, of people, and of machines is called "automatability" by study personnel. As contrasted with "automated", it implies that operations personnel, rather than design personnel, define the segments to be automated and the level of automation. This allows systems operation to evolve out of operational experience, for efficiency and reliability.

Figure 4-2 depicts GSFC POCC system evolution correctly as an endless circle. The evolutionary development and operation of new systems must come about by endlessly evaluating and modifying existing systems and creating compatible new

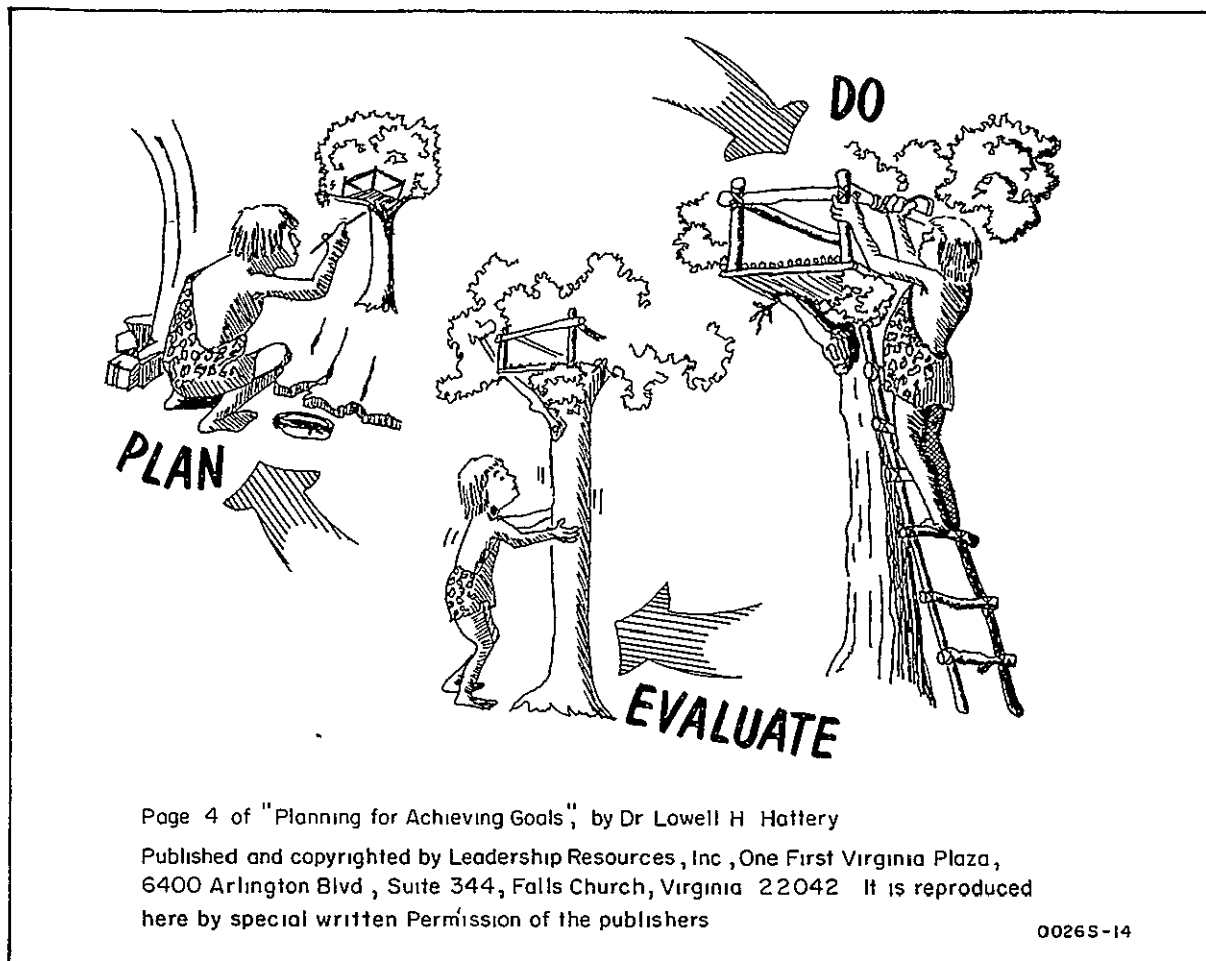


Figure 4-2. Endless POCC Systems Evolution

systems to meet emerging requirements. Therefore, Pocnet is driven by evolutionary organization and operation. Appendix 4E, Evolutionary Distributed System Design, discusses this concept more fully.

4.2.2 SYSTEMS CONCEPT

Figure 4-3 shows an overall view of how the systems concept was synthesized from external requirements and systems drivers.

4.2.2.1 Analysis of External Requirements. From the external requirements, it is evident that GSFC will maintain POCCs in the future, but they will include an increased number of more standard payloads and requirements. There will be up to twice as many payloads to support, with considerably higher potential data rates. On the other hand, the spacecraft, launch vehicle, and data distribution systems will become more standard. The functions to be performed will be more complex than the average GSFC POCC today but not more complex than some of the current POCCs such as ATS-6, IUE, Landsat, and SMM.

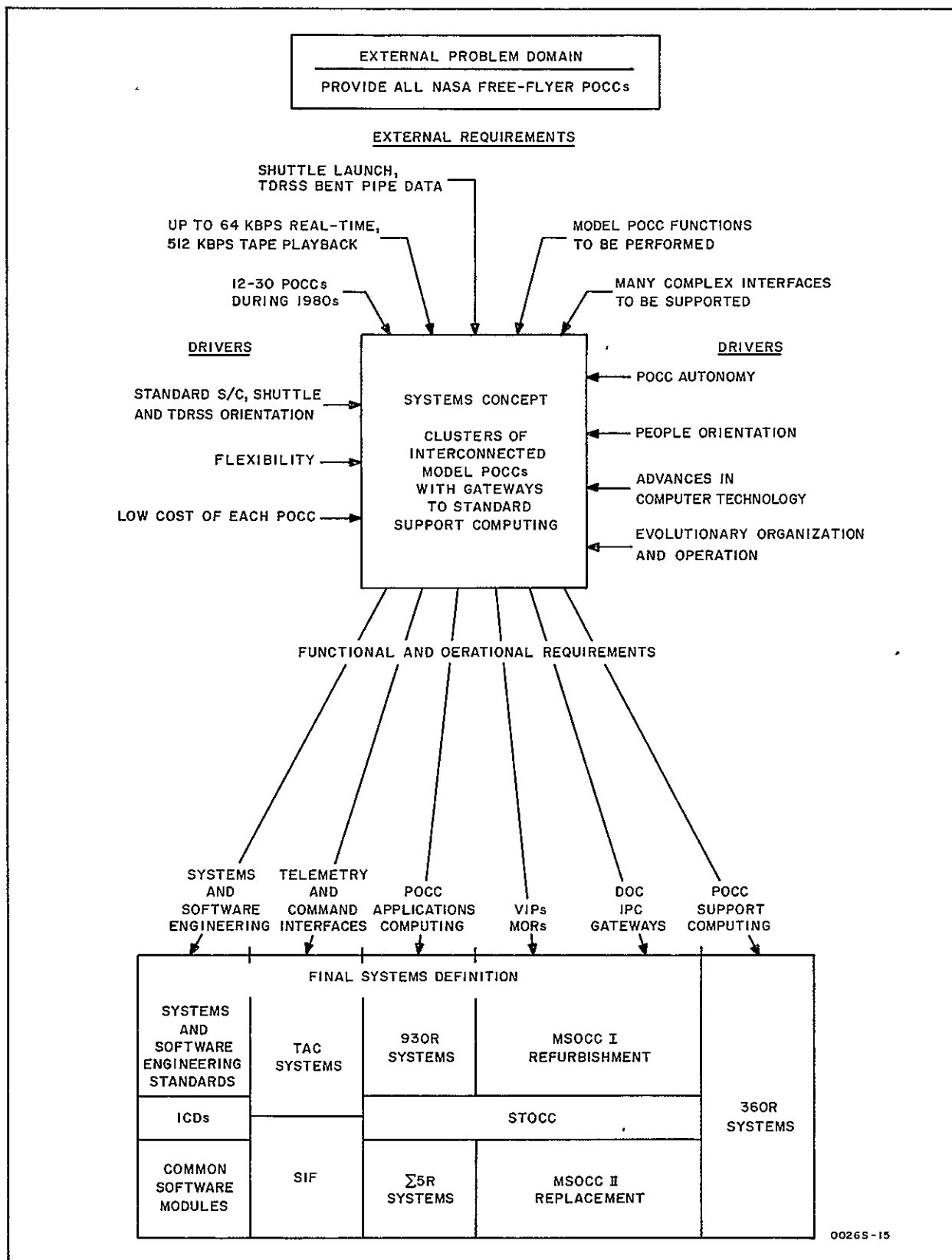


Figure 4-3. Systems Concept Synthesis

This implies that we should think in terms of supplying a larger number of POCCs which are essentially similar to those we have today, but the average POCC will have to handle higher data rates and will be more complex. At the same time we can think in terms of standardization of POCCs, but we have to recognize the need for an increased number of increasingly complex interfaces.

This tells us that we should be developing standard POCCs, but we should isolate the POCC external interfaces from the POCC internals as much as possible in order to maintain the standardization.

4.2.2.2 Analysis of Systems Drivers. The drivers imply other characteristics necessary for POCC standardization. Standard spacecraft support implies standard software which can be customized via unique data bases for each payload. Therefore, software engineering, common software modules, and data base management facilities must be provided within the concept. Shuttle orientation is a complex telemetry and command and operational interface for the first few days of a mission, followed by a quite different TDRSS bent-pipe interface. This implies that these characteristics should be isolated from the POCC internals.

Flexibility requires that the POCC be constructed of functionally independent subsystem pieces which can easily be taken apart and put together again for each payload mission or mission phase. The low cost of each POCC is then obtained by making these pieces compatible and standard.

POCC autonomy requires that each POCC hold and control its own hardware, software, and operational interfaces during the period of time it is controlling the payload, so that nothing which happens in other POCCs affects it.

People orientation demands attention to interfacing devices and languages, and this, in turn, means that an increasing percentage of POCC computing capacity must be allocated to terminal device support and user language processing.

Advances in computer technology imply two features. First, computing price/performance can be obtained either with mass production minicomputers or with high-end large systems (370-168) class). This gives two economical ways of implementing POCCs, either with dedicated applications processors or by time-sharing on a large system. Second, computer interconnection via international standard serial protocols running in \$1000 intelligent channels will be the way to interconnect computers in the 1980s and must be factored into any systems concept.

Evolutionary organization implies that GSFC should continue the current POCC concept of providing dedicated support for a required time period (whether for an hour at a time as in MSOCC or several years as in IUEOCC). Future systems should be an outgrowth of current systems rather than adopting an entirely new approach, for the current approach has worked well and has much improvement potential remaining.

Evolutionary operation implies that any systems concept should feature simple basic systems operational modes rather than all new complex or automated approaches. In addition, the systems operation should be able to easily evolve into complex and semiautomated modes as operational experience dictates. Design of evolutionary distributed systems is discussed in detail in Appendix 4E.

Figure 4-4 summarizes the application of the drivers to the synthesis of a systems concept.

The POCC autonomy issue together with management and operational complexity militate against the large centralized processor approach to GSFC POCCs in the 1980s. Although these large processors can deliver price/performance comparable to the minicomputers, the latter, organized into more or less conventional Model POCCs, have the considerable advantage of allowing the evolutionary continuation of existing GSFC POCC capability. Furthermore, the centralized processor would present potential for congestion, data bottlenecking, and interfacing problems which can be avoided with individual freestanding POCCs. Therefore, the individual autonomous POCC approach was adopted without detailed study of centralized approaches.

4.2.2.3 Synthesis of Systems Concept. The starting point for a POCC systems concept for GSFC is to take a standard minicomputer and dedicate it for a period of time (an hour or a year) to run the POCC applications. The applications processor (AP) hosts all the POCC applications for a single given payload for the period of time it is assigned. The difference between so-called dedicated POCCs (such as IUE) and so-called multisatellite POCCs (such as MSOCC) lies then in the length of time these APs are dedicated to a particular payload at any one time. The drivers of standardization, flexibility, and low cost imply that there should be a number of identical APs at GSFC, each one able to run standard command software customized by a data base for a particular payload.

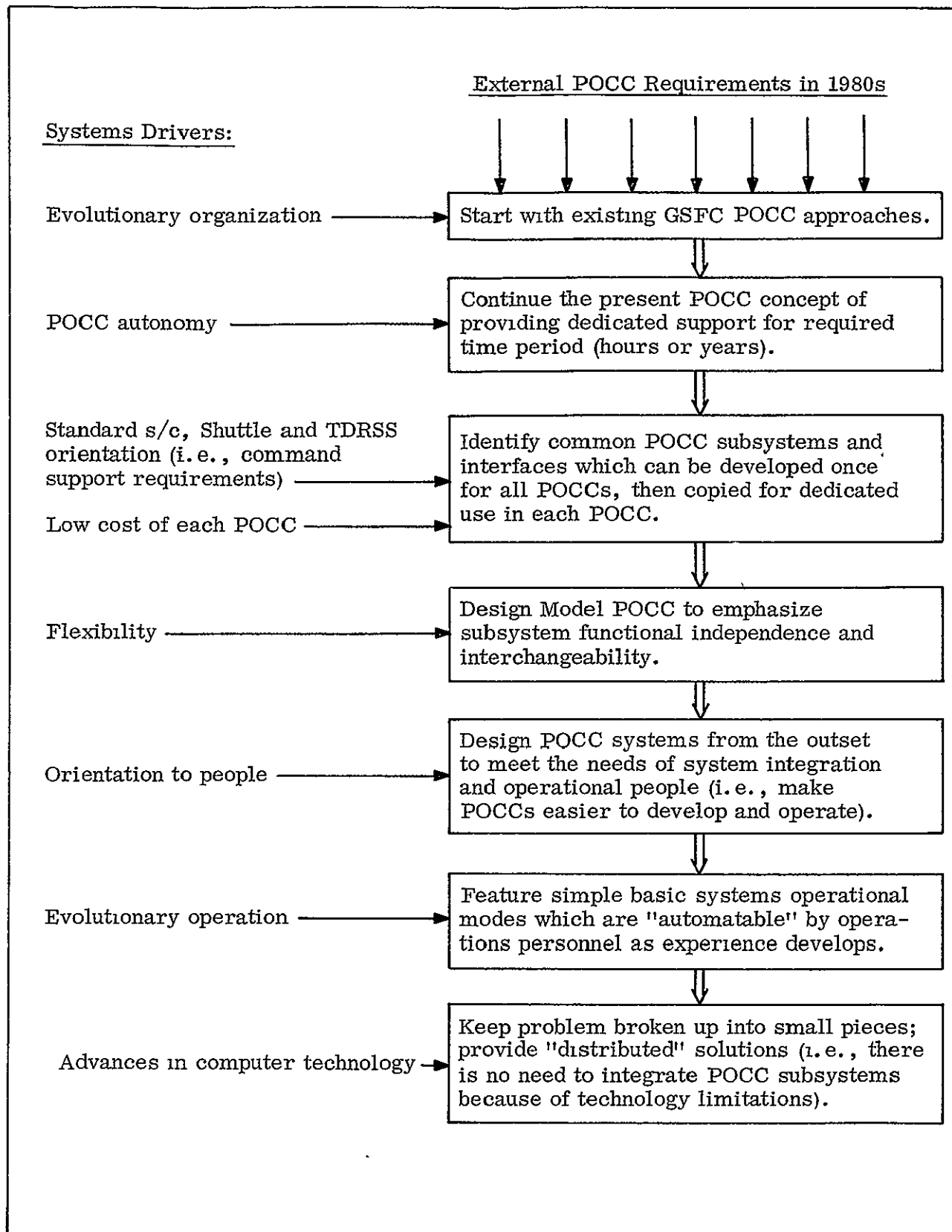


Figure 4-4. Summary of the Application of Systems Drivers to the Synthesis of a Systems Concept

Given the allocation of a POCC to be supported on a standard minicomputer, the issue of how to deal with interface complexity arises. The clear answer from a manageability point of view (if we can afford it from a hardware and software point of view) is to isolate nonstandard interfaces into "virtual interface" processors. In this way, each POCC essentially consists of a core of standard applications software running on a standard AP and surrounded by other processors which handle interface idiosyncracies and present a standard interface to the AP.

The following are the main interfaces to the AP broken into natural groupings:

- a. NASCOM interface.
 - (1) STDN (T&C).
 - (2) Shuttle MCC.
 - (3) KSC.
 - (4) Spacelab POCC.
- b. Mission Operations Room.
 - (1) Keyboard/CRTs.
 - (2) Command panel.
 - (3) Hard-copy unit.
 - (4) Strip-chart recorder.
- c. External systems.
 - (1) Mission support computers.
 - (2) Telops.
 - (3) External experimenters.
 - (4) NCC.
 - (5) DAS.

The natural decomposition into these groupings allows these interfaces to be isolated into only three kinds of systems, each one functionally independent of the others. The systems are described in the following paragraphs.

4.2.2.3.1 TACs. The NASCOM interface comprises principally payload telemetry and commands on a STDN bent pipe. However, it also includes planning and operational traffic from SMCC, KSC, or Spacelab POCC during Shuttle or Spacelab pre-flight or flight operations. The name TAC (for telemetry and command system) has been adopted for the processor which isolates this interface.

4.2.2.3.2 VIPs. The Mission Operations Room (MOR) interface comprises principally the terminal peripheral devices used to provide the human-to-machine interface for payload operations and system control. Each point of human-to-machine interface requires some kind of device. Supporting real devices requires a log of special real-time interrupt handling, hardware I/O interfaces, and software device handlers unique to the particular device being supported. CRTs from two different manufacturers generally have different ways of interfacing. Restricting the devices to certain manufacturers and models is not realistic, since there is no standard to go to and peripherals are constantly being improved and upgraded.

Thus, to isolate the system details of the MOR (what particular devices the MOR contains, how many there are, who made them, etc.), the MOR interfaces have been isolated on a processor called a virtual interface processor (VIP). The single VIP interface acts as the in/out box for the MOR. Communications between the AP and the MOR are at the standard virtual terminal level as described in Appendix 4A, Virtual Terminals for Spacecraft Payload Operations. In this way, the standard applications software on an AP can always drive standard virtual terminal formats in the MOR, no matter what manufacturer's devices are interfaced there.

4.2.2.3.3 Gateways. The third grouping of interfaces which a POCC must access is the category comprising all other systems external to the POCC. These would include mission support computers (for orbit ephemeris data, command memory loads, mission plans), Telops (for quick-look access to tape dumps and instrument data), external experimenters, NCC, and DAS.

Each of these systems has unique characteristics which may change from time to time, and which, for the most part, are not really germane to the POCC implementation. Hence, for each of these systems, a standard gateway process is identified which will isolate the details of the system interface from the user POCC.

All POCCs will use the same gateway standard, although this may be implemented by standardized software, a common hardware processor, or some combination thereof. In this way, one standard of communication can exist and be controlled across all POCCs, governed by a single Interface Control Document (ICD) with the external system.

4.2.2.3.4 IPC. For POCC autonomy, the specific individual AP, TAC, and VIP assigned for a period of time should be patched together as shown in Figure 4-5. This enables the single configuration shown to be isolated from any failures which occur in any other system outside the POCC.

On the other hand, there are many reasons for requiring communications paths other than those shown in Figure 4-5. A TAC may be assigned the role of diverting or duplicating a stream of data to a second AP (for example, for science, support computing, or hot backup operations). An AP may be serving a dual role as mission planning system, requiring interfaces to a number of planning terminals, not just those in one MOR. An MOR terminal may require access to mission support services, such as to request a command memory load. Finally, every POCC has to be able to get at every gateway, requiring either operational patching (untenable), 30 channels on each gateway processor (untenable), or a generalized Inter-Process Communication (IPC) system. A generalized IPC system would be connected to every POCC computer at GSFC and would enable any such computer to talk to any other computer at any time. The model configuration for IPC and gateways is shown in Figure 4-6.

4.2.2.3.5 Data Base Storage. Standard POCCs will rely more and more on custom data bases rather than custom software to give them the unique personality required to control a given payload. Furthermore, an operating POCC builds a real-time data base of operational timeline, spacecraft telemetry values, and status and systems current configuration information.

These data bases reside on the POCC AP when the POCC is operational to give the POCC its required autonomy. However, when the AP fails during an operational episode or when it is necessary to change from AP to AP, it is required to transfer data bases from AP to AP. This may be done by physically carrying disk packs from AP to AP. With enough operational activity it becomes more manageable to have a

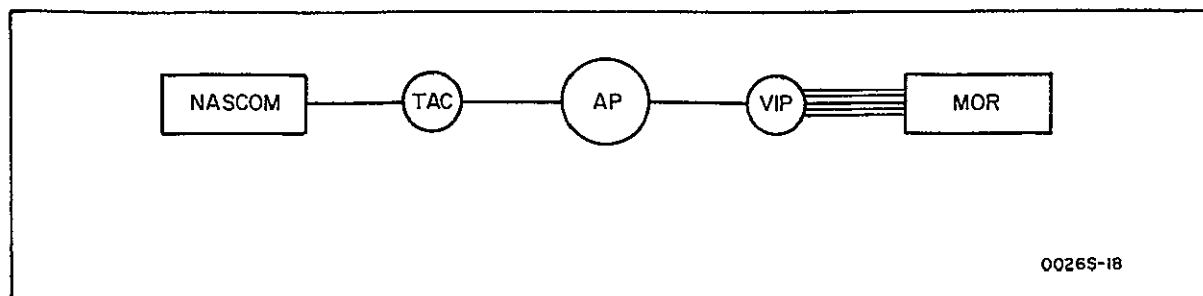


Figure 4-5. Autonomous Model POCC

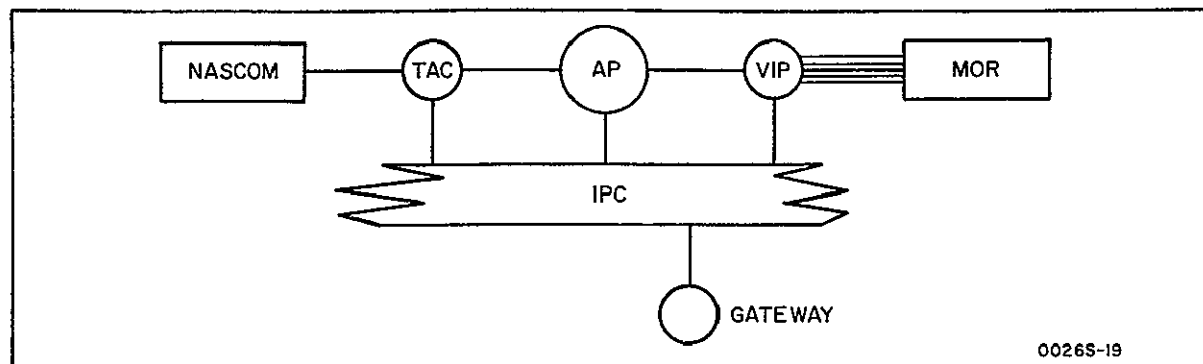


Figure 4-6. Fully Interconnected Model POCC

designated system for storing data bases between POCC episodes or during POCC episodes (for example, to bring up a warm backup AP quickly). In addition, a designated system is extremely desirable for hosting software engineering tools and providing configuration management services for all POCC systems development.

4.2.2.4 Summary. A systems concept has been synthesized starting from the existing GSFC POCC approach and responding to the external requirements of the 1980s and to the adopted generic systems drivers. This system concept, when fully implemented, would result in a distributed computing network of POCC subsystems. For this reason the systems concept has been named Payload Operations Control Center Network (Poccnnet).

4.2.3 SYSTEM REQUIREMENTS ANALYSIS

The external requirements from Section 3 are mainly at the functional system level in character and, hence, drive the functional decomposition into subsystems but not to the level required for performance specifications.

The hard numbers adopted for requirements for purposes of the study, extracted from Section 3, are as follows:

- a. Number of MORs: 12 to 30 for GSFC through the 1980s.
- b. Maximum real-time housekeeping rate: 64 kbps.
- c. Maximum tape dump or quick-look science rate: 512 kbps (SMM has a dump rate of 1024 kbps and ST has a quick-look science rate of 1 Mbps, but these are considered tall poles for POCC support and may require unique solutions).
- d. Number of terminal devices in one MOR: 4 to 20.
- e. Total number of terminal devices in Poccnet: 200 to 400.

The functional requirements of each system and additional performance requirements as appropriate are further developed within Sections 5 (Host Computer Systems) and 6 (Inter-Process Communication). These sections are not intended to represent specifications for the subsystems described, although in some cases considerable detail is developed. The requirements for many subsystems or components of subsystems in many cases depend on individual payload requirements or external system requirements (such as NCC), the development and formalization of which are outside the scope of this study. Indeed, the principal purpose of this study is to point out the systems and activities which must be pursued in order to accomplish the organizational goal of providing more standard POCCs and standard POCC support in the 1980s. For example, the development of TAC requirements and AP requirements in detail (from the general requirements presented in Section 5 and in the Concept document of Phase A) each involved over one man-year of activity beyond the scope of this study, and members of this study team participated in those developments. Other systems called out in this concept will be subjected to detailed scrutiny and requirements analysis as they become timely.

A Program Development Plan resulting from this study will detail the work breakdown structure required to continue the elaboration and development of Poccnet systems requirements and specifications and will present implementation phasing plans, schedules, and resources.

4.3 SYSTEMS STUDIES

Most of the Pocnet systems concept and requirements presented in paragraph 4.2, and elaborated in Sections 5 and 6, are well within the current state of the art of systems engineering. Further development would simply be a matter of applying the recognized systems engineering methodology, that is, detailed formal requirements, designs, implementation, and test.

However, two areas were deemed to require specific system-level study. One was validation of the IPC design approach chosen by the IPC team. The other was the selection of a programming language or languages in which to implement any systems which are acquired.

4.3.1 IPC SIMULATION

The IPC system was considered especially vulnerable to performance deficiencies. If for any reason the IPC system should become congested, real-time data would then back up and ultimately be lost as steady state queues increased without bound.

Hence, it was decided to develop a detailed computer model of the traffic which a number of simultaneous POCC operations would cause within IPC. This model was made as realistic as possible, including simultaneous real-time operations with payloads, together with traffic in and out of a designated data base storage system. Several simulations with different configurations were run based on a realistic model of the IPC PMP Subnetwork (refer to Section 6). The models, together with a description of the simulation approach and results, are given in detail in Appendix 4B, Pocnet Simulation Study (with Attachments). The results are summarized in the following paragraphs.

4.3.1.1 Summary of IPC Simulation Results. The principal finding of the simulation analysis was that it would be virtually impossible to overload the proposed IPC system with data traffic, even with unrealistically heavy workload models (for example, the simulated system handled three simultaneous 64-kbps real-time streams and a 512-kbps tape dump through one node with only a fraction of IPC capacity utilized).

With three 512-kbps dumps going simultaneously through one node, that node did become overloaded. However, this is an unrealistic loading because in the Model POCC

such tape dumps come through the TAC directly to the AP without passing through IPC at all. Hence, a tape dump going through an IPC node must be considered in the nature of a contingency rather than nominal. Furthermore, the degradation in the node was of a type which could easily be handled by faster memory or a private CPU bus.

Therefore, the IPC simulation study did in fact validate the required performance aspects of the IPC design approach for realistic POCC traffic models.

4.3.1.2 Further IPC Study Activity. Subsequent to the conclusion of the IPC study reported in Section 6, alternative IPC approaches promising somewhat reduced flexibility at lower cost are to be evaluated. These include incorporating the IPC system within the VIPs, a digital switch design from Data General, a cable bus design based on video cable technology from CSC Systems Division, and the Shuttle MCC Multibus from Ford Aerospace.

Simple crossbar switches, most digital buses (such as the Network Systems Corporation Hyperbus), and common memory (such as the KSC LPS command data buffer) were ruled out during the study because of special interfacing requirements or lack of flexibility. (Refer to Section 6.)

The nominal design concept described in Section 6 will be further prototyped on paper and then in hardware and software, if necessary, to establish performance and cost definitively. When this information is available, together with the further studies of the remaining potential low-cost candidates mentioned above, an IPC design approach will be chosen and specifications for development will be prepared.

4.3.2 SYSTEMS IMPLEMENTATION LANGUAGE STUDY

A substantial system-level study was performed in the area of programming languages for implementing Pocnet systems software. This study, directed by Dr. Victor Basili of the University of Maryland, developed objective criteria for evaluating possible Pocnet implementation languages and compared 15 existing languages on the basis of these criteria. The languages examined were the following:

- a. BLISS-11 — A systems implementation language for the PDP-11 series.
- b. C — The language of the UNIX operating system.

- c. Concurrent Pascal — A high-level language for writing operating systems.
- d. CS-4 Base Language — An extensible language being developed for the Navy.
- e. FLECS — A Fortran preprocessor.
- f. HAL/S — The NASA language for the Space Shuttle program.
- g. Interdata Fortran V — An extension of ANSI Fortran.
- h. JOSSLE — A PL/I derivative for writing compilers.
- i. JOVIAL/J3B — A close relative of JOVIAL/J3, the Air Force standard language for command and control applications.
- j. LITTLE — A Fortran derivative that operates on bit strings of arbitrary length.
- k. Pascal — A highly structured, general-purpose language.
- l. PREST4 — A Fortran preprocessor.
- m. SIMPL-T — The base member of a highly structured family of languages.
- n. SPL/Mark IV — A high-level language with many machine-oriented features.
- o. STRCMACS — A collection of structured programming macros for IBM OS/360 assembly language.

Each of the languages was characterized by its syntactic features, machine dependence, efficiency, level of the language, size of the language and compiler, special system features, error checking and debugging, design support (modularity, modifiability, and reliability), and the use and availability of the language. These characteristics were then rated against a list of Pocnet requirements for general systems programming, real-time processing, data base management, numerical processing, and data formatting and conversion.

The entire study is attached to this report as Appendix 4C, A Study of Systems Implementation Languages for the Pocnet System. A brief summary of the principal recommendations is given in the following paragraphs.

4.3.2.1 Summary of Consultant's Recommendations. None of the languages studied satisfies all the requirements for Poccnet systems implementation. Therefore, consideration should be given to the language which satisfies most of the requirements at lowest cost. Costs include start-up cost, development and testing costs, and maintenance costs.

The family of languages approach, using either of the Pascal or SIMPL families as a base, was recommended as the first alternative. This approach would provide a small nucleus language, together with extensions for special purposes such as communications.

The second alternative would be to use a single language that meets most of the requirements. CS-4, HAL/S, JOSSLE, JOVIAL/J3B, and SPL/Mark IV lie in this category. HAL/S was recommended as the best choice among these five.

The third alternative was Fortran. It failed on many counts, and the consultant was unable to come up with any enthusiasm for it other than its widespread use. If Fortran is chosen over his objections, he recommended that a "Structured Fortran" preprocessor be selected to provide control structures for structured programming. Of the two preprocessors studied, FLECS was considered the better choice over PREST4.

The fourth alternatives were a group considered too low level for general use in Poccnet (such as BLISS-11 and LITTLE) or having difficult-to-read syntax (such as C).

Finally, assembly language was also considered too low level for general use. For special cases where time or space efficiency is critical, the consultant recommended that a set of structured programming macros be used.

4.3.2.2 Study Manager's Recommendations. The study manager agrees in the main with the conclusions of the consultant and believes that a valid technical assessment of the languages and requirements was made. However, following the consultant's report, additional considerations became apparent which substantially changed the picture, in the opinion of the study manager.

The principal recent development is that the Department of Defense (DOD) Common High-Order Language (HOL) activity (Figure 4-7) is moving into the prototyping phase,

Can Government 'Clout' Lead to DOD-1 Acceptance?

By Robert L. Glass

Special to Computerworld

WASHINGTON, D C — Can the clout that created Cobol gain acceptance for a new, general-purpose engineering applications language?

The Department of Defense thinks so. It is well into the process of defining just such a language, to be called DOD-1, and the current expectation is that the language will be rigorously defined and a test translator for it implemented by early 1979.

(It was the DOD which gave Cobol its shove toward widespread acceptability when it required Cobol compilers to be available for all computers procured for military inventory in the late 1950s.)

Groundwork for the new language, laid over the past couple of years, has included the following steps:

- Issuance of a DOD directive in April 1976 (number 5000-29) which, among other things, requires the use of a DOD-approved high order programming language (HOL) for all defense system software, unless it can be demonstrated that use of a HOL is either not cost-effective or not technically feasible. The goal is to force use of approved HOLs rather than assembly language or unusual HOLs.

- Iteration on a series of definitions of the requirements for a common DOD HOL. The requirements, reviewed by military, academic and industrial computer experts internationally, have progressed from the original "Strawman" proposal in early 1975, to a still-tentative "Woodenman" phase in August 1975, to a more orderly "Tinman" release in March 1976, and finally to a compact and stable definition in "Ironman," dated January 1977.

facilities and proofs of correctness, extensive module libraries and even semiautomatic programming from specifications," he said, pointing out that currently "the average programmer's tool box is rather bare."

Non-Military Usage

Whitaker sees potential for the use of DOD-1 outside the military. "Its use in DOD, and the provision of tools by the DOD would make it a popular candidate for use elsewhere." This has not happened with military languages in the past, he said, because there were too many of them.

Goals of the new language, according to Whitaker, include reduction in life cycle software costs, transportability of application code, improving maintainability and reliability and the capability for extremely efficient compiled code.

It is the intent of the DOD that the new language also be modern in concept. The

base language which will be focused upon in the development of the language specification will be either Pascal, PL/I or Algol 68, according to Whitaker. None of the languages on the interim list is sufficiently modern to be considered a good starting point. Thus DOD-1 is assured of being different from existing DOD languages.

Requirements in the "Ironman" specification include such advanced concepts as strong typing (the requirement for user declaration of data types, with compiler type consistency checking), encapsulated definitions (clusters of data with access restrictions), structured programming control structures (plus the ubiquitous GO TO), and parallel processing.

In addition, more traditional concepts such as the following will be provided: integer, scaled, fixed, float, character and Boolean data types, array (homogeneous) and record (heterogeneous) data ag-

gregates, functions and procedures (including recursion), input/output, and exception handling (including specification of assertions which, when false, invoke an exception).

One surprising result of the DOD-1 program to date has been the commonality of requirements across application lines. Military user communities typically divide into such categories as avionics, weapons systems, guidance, command and control, communications, training simulators, etc. "It was impossible to single out different sets of requirements for different communities," Whitaker said. "Requirements solicited from these diverse users were identical."

The expectation is that as soon as DOD-1 becomes available, most languages on the interim list will be phased out. Existing code is, of course, exempted from both the interim list requirement and its phaseout, Whitaker noted.

- Establishment of a list of allowable interim HOLs which may be used until DOD-1 is available. The list consists of CMS-2 and SPL-1 (Navy languages), Tacpol (an Army language), Jovial J3 and J73 (Air Force languages) and Cobol and Fortran. Each language on the list is to be assigned to a governmental control agent to prevent deviations from standard definitions.

- Signing of a contract within the next month for a definition of the language (guided and constrained by the "Ironman" requirements) and a pilot implementation. Lt Col William A. Whitaker, Air Force officer in charge of the DOD-1 process, said the goal of DOD-1 is to take a more orderly

approach to the \$3 billion that DOD spends annually on computer software. Whitaker, who works at the Defense Advanced Research Projects Agency in the Pentagon, calls the advantages of HOL use "compelling."

A common DOD HOL, he said, will not only provide the traditionally well-known HOL advantages of reducing programming costs, increasing maintainability and providing some measure of application software portability, but it will also allow the development of associated modern language-dependent software support tools.

"A total programming environment includes not just compilers and debugging aids, but text editors and interactive programming assistance, automatic testing

00265-20

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR.

Figure 4-7. Computerworld Article on Government Acceptance of DOD-1

with several contracts placed in mid-1977 to develop language designs. The 5-year budget for this activity considerably exceeds the entire Pocnet 5-year budget. A selection and formal adoption of a standard DOD HOL before the end of 1979 seems nearly a foregone conclusion.

This DOD Common HOL would be very similar to Pascal and HAL/S, based on public statements made by the Chairman of the activity. Pascal and HAL/S were very highly rated by the Pocnet language consultant (refer to paragraph 4.3.2.1), the main drawbacks being lack of widespread public use or lack of a suitable standardization authority. The DOD software budget is greater than the entire NASA budget, hence, DOD "clout" should provide acceptance and DOD institutionally will provide the standardization authority. Therefore, the study manager has concluded that Pocnet should wait for the DOD Common HOL.

In the meantime, it is recommended that the consultant's remaining recommendations be followed. This means that Pocnet software should be developed in Fortran where possible and macro assembly where necessary, using structured programming processors for both languages.

4.4 SOFTWARE ENGINEERING

Software engineering covers a range of activities related to the development of high-quality software systems. It involves the skillful application of software development tools and methods based on a sound understanding of basic principles. This new discipline is concerned with the use of these tools, methods, and principles to economically produce software which will run reliably on today's computers. Thus, software engineering is a practical discipline rather than an academic one.

The computer hardware used in control centers, past and present, as well as that which will be used in the POCCs of the 1980s, is designed and constructed according to time-tested standards in the fields of electrical engineering, mechanical engineering, and manufacturing. By contrast, most POCC applications software developed up to this time has not been produced according to an accepted set of standards simply because no such standards existed. To correct this situation, and thus lay the groundwork for the development of cost-effective, reliable, reusable software, a study of software engineering methodologies was made as part of the Pocnet definition phase study. As a result of this effort, a recommended

standard approach for software engineering was developed and documented. This approach is described briefly in paragraph 4.4.1 and in greater detail in Appendix 4D, Poccnet Software Engineering Standard Approach Recommendation.

A related area of the study is the on-going activity of the Common Software Steering Group (CSSG). The current state of this activity is described in paragraph 4.4.2.

4.4.1 SOFTWARE ENGINEERING STANDARDS

Years of experience in implementing POCCs at GSFC have resulted in a good understanding of the development and operating problems of POCC computing systems. Under current techniques of implementation, transfer of this experience from one system to the next is slow and uncertain. More portability of both hardware and software will be essential in the future.

In addition, the increasing complexity of control center applications, involving real-time attitude determination, data and program management for spacecraft on-board computers, graphics displays, spacecraft simulations, and more is beginning to increase integration and test time to unmanageable proportions. A better approach to software engineering is needed.

Because of the large number, variability of duration, and variety of POCCs to be developed in the 1980s, a flexible approach has been developed which can be tailored to each individual project's needs. This approach is documented in Appendix 4D.

Poccnet is an evolutionary system designed to adapt to new requirements. This characteristic makes the use of a formal engineering approach essential to maintain system integrity during constant change. A single, unified methodology is also essential to allow communication among the wide variety of Poccnet users, including applications programmers, systems programmers, experimenters, and mission controllers.

The software engineering approach described in Appendix 4D is designed to be applicable to every project contributing to Poccnet. This approach is made possible by specifying a generalized development plan which will be custom tailored under the supervision of a Software Engineering Panel (SEP) to the particular requirements of each project. This methodology will also provide a standard vocabulary to enhance communication among all involved organizations. Finally, the

approach takes advantage of the best current techniques for developing modular, reliable software and for effectively managing development. This is consistent with the NASA Software Management Guidelines, which specify a phased software life cycle with accompanying configuration management and quality assurance activities.

The organization and assignments of responsibility for the SEP are described in the following paragraphs and illustrated in Figure 4-8.

The Software Engineering Manager (SEM) acts as chairman of the SEP and will be responsible for maintaining and updating this standard engineering approach. The SEM will also be responsible for interpreting the standards and applying them to individual projects.

The Quality Assurance (QA) group reports to the SEM and is responsible for technical review activities for developing systems. Typical activities include reviewing test plans and reports, reviewing source code and documentation for conformance to standards, and reviewing system designs for adherence to the standard methodology.

The SEP is chaired by the SEM and is composed of individuals knowledgeable in software engineering and familiar with the standard approach. Panel members must also understand the development problems of individual Pccnet projects. Specific responsibilities of the panel include:

- a. Ensuring that the methodology is adhered to.
- b. Interpreting the methodology when questions arise.
- c. Overseeing the training program.
- d. Changing the methodology when necessary because of problems arising in the application of the methodology in unusual or unforeseen conditions.
- e. Incorporating new techniques as they are developed and become applicable.
- f. Recommending and securing new automated tools.
- g. Overseeing the collection of data and developing and evaluating metrics to be used in further refining the methodology.

The standard approach is summarized in the following paragraphs and illustrated in Figure 4-9. The interested reader should refer to Appendix 4D for details.

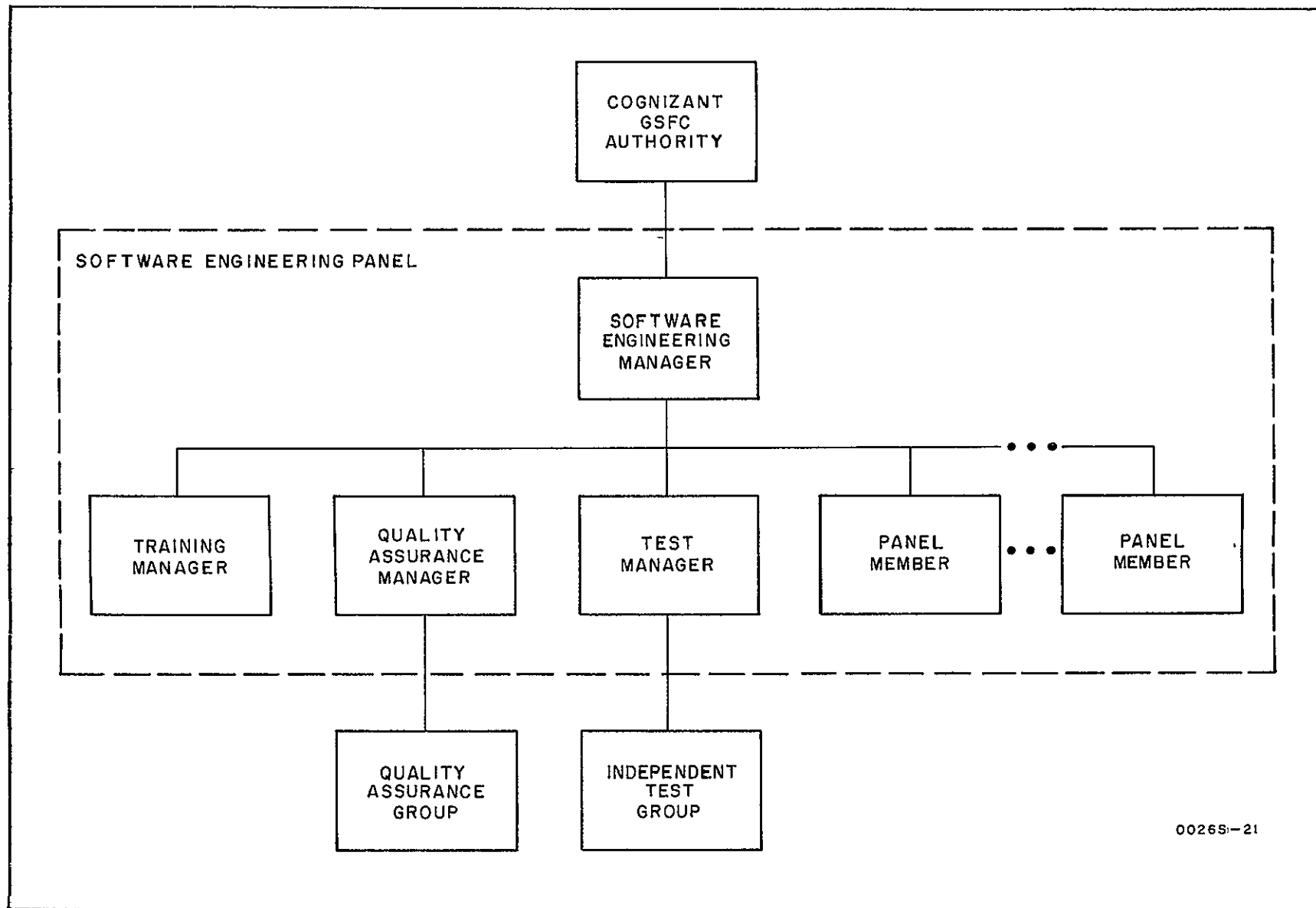
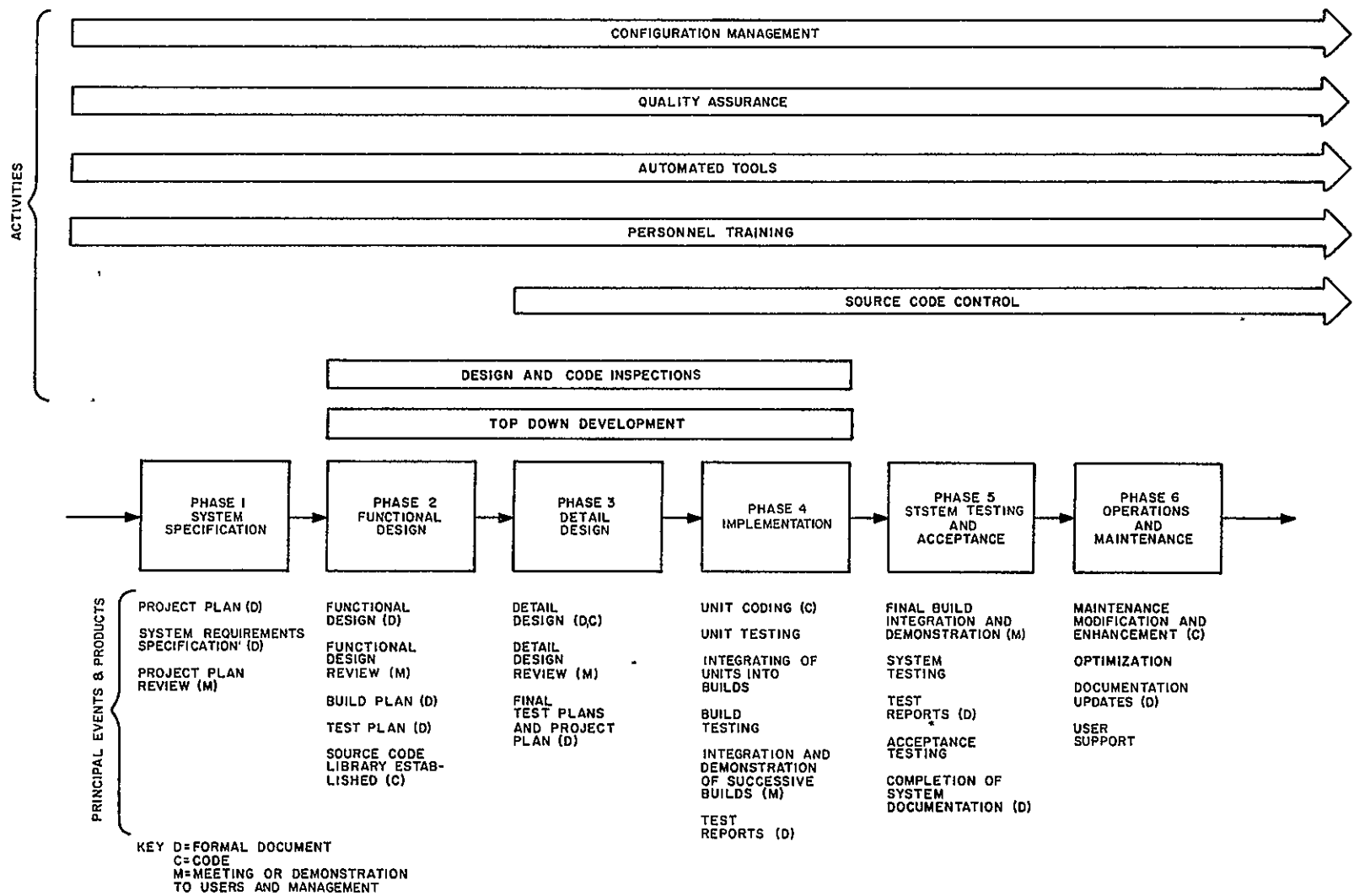


Figure 4-8. Recommended Composition of Software Engineering Panel

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR.



00265-22

Figure 4-9. Standard Approach Overview

The basis of the standard approach is a six-phase system life cycle consisting of system specification, functional design, detail design, implementation, system testing and acceptance, and operations and maintenance. Each phase has well-defined activities and end products. The intent of defining distinct development phases is to increase management visibility and control and to facilitate configuration management.

The rest of the standard approach describes specific activities which take place either within certain phases or across many or all phases. For example, documentation standards apply to every phase of the life cycle.

In Figure 4-9, below each phase are listed the principal events and products of that phase. These are milestones for measuring progress and for configuration management activities. The upper part of the figure shows other important elements of the standard approach. The bars indicate the beginning and duration of each activity.

Considerable emphasis is placed on the first three phases, since it has been shown that, if requirements specifications and designs are clearly stated, integration and testing problems are minimized. By the time of system testing and acceptance, many, if not most, of the system capabilities will have already been verified and there will be considerable confidence in the ability of the system to perform as originally intended.

Section 3 of Appendix 4D describes an approach for effective software engineering consisting of top-down development, modularity, design and code inspections, and staged implementation. This approach structures the software system as a functional hierarchy, with defined criteria for modularization and interprocess communication. Design generally proceeds from the user-visible interfaces down to the least visible inner components of the system.

Formal inspections are required to be conducted for both the designs and code. These inspections will help ensure that errors or oversights are not designed or coded into the system in the first place.

Systems are required to be implemented in incremental stages, so that system functions will become visible early in the development effort, rather than all at once at the end of development. Validation, testing, and documentation activities proceed concurrently with development.

Documentation must be a product of every Poccnnet development effort. This implies that any standards for documentation must be flexible enough to be applicable to a wide variety of types of documentation. The standard approach allows document outlines to be tailored to particular needs of projects yet remain within a common framework. Emphasis is placed on identifying the audience for a document and providing material which is useful to that audience. Detailed subroutine specifications will be in machine-readable form. Formal documentation will supplement, not duplicate, machine-readable documentation produced during design and coding.

Use of computers to support the development process is essential to current and future projects. This is basically a matter of economics, since manpower is now the single most expensive aspect of software production. As a result, any use of computers to save human time and energy is a great economy. Software tools can be used by managers and system users as well as by programmers. The standard approach describes the types of software tools, including programming languages, text editor, programming support library, debugging tools, simulators, analyzers, and configuration management tools, which should be made available to Poccnnet development projects.

Computer technology is changing fast. As a result, both technical staff and management must be continually informed of new techniques and educated in their use. The methods that sufficed to develop software in the last two decades are no longer feasible within today's limited budgets. Although new methods have been demonstrated to be feasible and economically more productive, they have not been adopted generally since it takes effort to change, and there is always resistance to the new and unknown. If Poccnnet is to be produced using the best available methods, educating both management and technical staff about these methods is essential.

Continual management review of developing systems is essential if they are to be developed on time and within their budgets. The QA group assists management by performing technical evaluations of the developing software and by maintaining programming and documentation standards. Standards are necessary to ensure that the delivered software will be maintainable, compatible with other Poccnnet software, and portable from one application to another.

Configuration management techniques (also known as change control), used for years in hardware engineering, are being successfully applied to software engineering.

The intent of configuration management procedures is to ensure that the developing system is always well defined. Without configuration management, controlling the specifications and implementing a large software system are extremely difficult.

Configuration management procedures are applied throughout the system life cycle. The procedures specified in the standard approach begin with the system specifications and carry through the design phases, implementation, system testing and acceptance, and operations and maintenance. At any time it will be possible to determine what the system consists of and what it is capable of doing. Effective configuration management eventually depends on control of the actual computer source code. General procedures are described for controlling access and changes to the code and for identifying the code releases.

Activities which were previously lumped together and referred to as "testing" have been analyzed and defined. This definition differentiates validation activities from verification activities and points out that these activities are not limited to the system testing and acceptance phase. Validation and verification activities begin during the design phases and are especially concentrated during system integration. The approach describes the differences between validation and verification and gives guidelines for test planning.

4.4.2 COMMON SOFTWARE STEERING GROUP

The Common Software Steering Group (CSSG) is a committee composed of representatives from the Multimission Modular Spacecraft (MMS) Project and operational support systems elements within the Mission and Data Operations Directorate. Its purpose is to promote software commonality, principally for MMS. CSSG functions are as follows:

- a. To coordinate software requirements, design, and implementation.
- b. To provide systems management visibility of software functional specifications, interfaces, and milestones.
- c. To develop and set guidelines for software engineering and mathematical definitions and algorithms.
- d. To identify systems and operations concepts which promote software commonality in future systems.

- e. To work with functional management in developing policies which favor standard systems and common software.

The primary goal of the CSSG is to identify and specify a set of important, widely useful common software modules to be developed and made available in time to support Solar Maximum Mission and future MMS users, together with guidelines for systems design to promote the common software.

Three subgroups of the CSSG were formed to study three specific areas: software engineering standards, mathematical analysis, and software and ground systems software. The needs, objectives, and near-term goals of each of these subgroups are discussed in the following paragraphs.

4.4.2.1 Software Engineering Standards Subgroup. The principal needs in the software engineering standards area are two-fold:

- a. Establish an overall discipline under which common software evolves.
- b. Establish a comprehensive mechanism for the gathering, storing, and retrieving (for dissemination to prospective users) common software products.
- c. Ensure that required automated software engineering tools are acquired and used.

These needs evolve from the observation that the software which comes under the aegis of CSSG should be common, not only by virtue of its generally applicable use but also by virtue of its overall development and availability. Software bearing the CSSG seal would be, above all, useful and necessary. For this reason, CSSG software must be developed to standards of very high quality, modularity, and ease of use and must be made easily available to those who need it.

These needs give rise to the following two principal objectives:

- a. Adopt a set of software engineering guidelines which affect all aspects of the software development life cycle and which ensure the necessary quality and other characteristics of software bearing the CSSG seal.
- b. Establish a repository for common software and support tools which contains both programs and associated documentation, consistent with recommendations made in the guidelines.

The near-term goals of this subgroup are as follows:

- a. Evaluate currently existing or recommended software engineering approaches (Pocnet, MMS, etc.), standards, and tools recommended by the M&DOD Software Engineering Lab, and issue a set of CSSG Software Engineering Guidelines.
- b. Lay out a structure for the common software repository activity which addresses the following issues and requirements:
 - (1) Classification of the common software products.
 - (2) Cataloging procedures and journaling.
 - (3) Degree of automation of the repository.
 - (4) Final validation and verification procedures.
 - (5) Trouble reporting and software maintenance.
 - (6) Overall management and maintenance.

4.4.2.2 Mathematical Analysis and Software Subgroup. The needs identified in the area of mathematical analysis and software are to eliminate redundant software development efforts, to ensure consistency of results of calculations performed by separate support system elements, and to ensure ease of communication between the separate elements (such as common coordinate system definitions). To meet these needs, the following objectives were adopted:

- a. Identify and specify a standard planetary ephemeris source, and work toward a standard ephemeris representation based on that source.
- b. Identify and specify a standard star catalog system.
- c. Identify candidates and work toward selection of standard orbit tape and orbit file representations and standard orbit generators for support system use.
- d. Specify standard sensor models for MMS.

4.4.2.3 Ground Systems Software Subgroup. The principal need in the ground systems software area is to standardize the design of the software systems. This

is to ensure that systems are functionally modularized in common ways with well-defined interfaces. Implementations can then be developed on different systems, with implementers choosing what they can use of the common software and developing their own unique software. The design commonality would ensure that the system fits together properly.

This need imposes the following rational, achievable objectives:

- a. Choose important, inherently common key functional elements (a candidate list was developed).
- b. Concentrate on developing good modular designs for these elements.
- c. Do not try to do too much, and be sure that whatever is done is right and on schedule.
- d. Identify areas of the total ground system which can affect software standardization and provide guidelines to systems designers to promote standardization.

This subgroup has developed the following near-term goals:

- a. Develop a payload data base for MMS/SMM.
- b. Specify a Standard Systems Test and Operation Language (STOL) for payload ground systems.
- c. Develop a baseline set of control center standard software together with a programmer's manual for its use as part of the 930R procurement.

Progress has been made toward reaching these goals. A preliminary design for the MMS/SMM payload data base has been developed and documented in GSFC X-408-77-116, Payload Data Base Concept (Appendix 3B). Syntax and semantics of a baseline STOL have been developed and approved by the STOL configuration control board (CCB). This effort is documented in GSFC-X-408-77-100, GSFC STOL Functional Requirements and Language Description. The 930R standard software is currently under development.

SECTION 5
HOST COMPUTER SYSTEMS

SECTION 5

HOST COMPUTER SYSTEMS

CONTENTS

<u>Paragraph</u>		<u>Page</u>
5.1	General	5-1
5.2	Host Computer Capabilities	5-2
5.3	Applications Processors	5-5
5.4	Data Base Storage System	5-8
5.4.1	Introduction	5-8
5.4.2	Objectives	5-10
5.4.3	DBS Functional Overview	5-11
5.4.4	DBS Hardware Configuration	5-17
5.4.5	Processor Coordination	5-18
5.4.6	Backup and Recovery	5-21
5.4.7	DBS System Software	5-22
5.4.7.1	Vendor-Supplied Operating System	5-22
5.4.7.2	File Management System	5-23
5.4.7.3	Data Base Management System	5-23
5.4.7.4	Pocnet-Developed Systems Software	5-23
5.4.7.5	Software Tools	5-24
5.5	Virtual Interface Processors	5-25
5.5.1	Requirements	5-25
5.5.1.1	Need for Virtualization of I/O Devices	5-25
5.5.1.2	Number of Virtually Interfaced Devices Required	5-25
5.5.2	Functional Definition of Virtual Terminals	5-26
5.5.2.1	Random Access Block Display	5-26
5.5.2.2	Size System	5-27
5.5.2.3	Notational Convention	5-28
5.5.2.4	Text and Commands	5-28
5.5.2.5	Complete Commands	5-28
5.5.2.6	Positioning Controls	5-29

CONTENTS (Cont)

<u>Paragraph</u>		<u>Page</u>
5.5.2.7	Elements Common to All VT Commands . . .	5-30
5.5.2.8	MODESET Commands	5-30
5.5.2.9	SENSE Commands	5-31
5.5.2.10	PUT Commands	5-32
5.5.2.11	GET Commands	5-33
5.5.2.12	Object Coding of VIP Commands	5-33
5.5.3	Virtual Users -- Functional Definition	5-33
5.5.4	Host Requirements for VIP and Gateway Proc- essors	5-35
5.5.4.1	Multiprogramming and Reentry	5-35
5.5.4.2	String Handling	5-36
5.5.4.3	Control Blocks, Buffers, and Working Storage	5-36
5.5.4.4	Microprogramming	5-36
5.5.4.5	I/O Channels	5-36
5.5.4.6	Timing	5-36
5.6	Gateways	5-36
5.6.1	General Characteristics	5-36
5.6.2	Functional Description of Gateways	5-37
5.6.3	Interfaces To Be Provided Initially	5-38
5.6.3.1	Telops	5-38
5.6.3.2	Johnson Space Center	5-38
5.6.3.3	System 360 at Goddard Space Flight Center . .	5-39

ILLUSTRATIONS

<u>Figure</u>		<u>Page</u>
5-1	Establishing Logical Connections in Poccnet	5-4
5-2	Applications Processors in Poccnet Environment . . .	5-6
5-3	Centralized and Decentralized Data Base Philos- ophies	5-12
5-4	Poccnet Data Base Philosophy	5-14
5-5	Proposed Structure of DBS	5-15

ILLUSTRATIONS (Cont)

<u>Figure</u>		<u>Page</u>
5-6	Pocnet DBS Subsystem Hardware Configuration	5-16
5-7	Overview of Scheduling	5-20
5-8	Virtual User Scenario	5-34
5-9	Gateway Data Paths	5-38

TABLE

<u>Table</u>		<u>Page</u>
5-1	Levels of Service	5-18

SECTION 5

HOST COMPUTER SYSTEMS

5.1 GENERAL

Pocnet is based on the approach of distributing modern minicomputers together with modern real-time multiprogramming operating systems to provide the process and file management services needed to run control center systems. These minicomputers are called host computer systems. Paragraph 5.2 sets forth the general characteristics of host computer systems. Subsequent paragraphs discuss the following specific types of host computer systems:

- a. Applications processors (AP), which support typical POCC applications processing, data base processing, and data operations control.
- b. Virtual interface processors (VIP), which support real input/output devices in such a way that they appear to be idealized virtual devices to the processes which use them.
- c. Gateway processors, which provide interfaces between Pocnet and the outside world.

The telemetry and command (TAC) processor, which will be the standard interface to STDN via NASCOM for telemetry and commands, is another type of host computer. The system concept for the TAC was presented in Section 4. Because the TAC is fully described in a preliminary specification (Telemetry and Command Systems, GSFC S-511-3, July 1977), it will not be discussed in detail in this report. However, the major goals of the TAC are set forth in the following paragraphs. . .

The TAC will have a channel external clock rate of up to 1.6 Mbps with an average telemetry input rate of up to 600 kbps and an average command output rate of up to 56 kbps. In operation, the TAC will receive NASCOM bent-pipe 4800-bit telemetry blocks, perform any required error detection/correction processing, and forward spacecraft-oriented data to the APs. It will also accept command messages from the APs, convert them to NASCOM format, and handshake them through NASCOM and any other intervening systems (such as Shuttle MCC) to the spacecraft. Each TAC will be capable of servicing up to three host computers over separate channels. Normally, one of these channels will service the POCC AP. Also, the TAC will be able to accommodate up to three payload data streams, data rates permitting.

The TAC will be able to run payload-unique applications programs to strip or modify telemetry, to detect alarm conditions, or to verify commands. The TAC will be able to use an optional disk storage unit to buffer high-speed data before sending it on to an AP at a slower rate. It will also provide the capability to support simulations of telemetry and command interfaces in order to check out POCC systems and train POCC personnel.

Another standard front-end system not discussed in detail in this document is the line/monitor/recorder (LM/R), which attaches to the line interconnecting NASCOM with the TAC. The LM/R is a simple block monitor and recording device which is capable of examining all NASCOM 4800-bit blocks transferred to the TAC, recording them (for example, during a TAC outage), and playing them back on demand (for example, following a pass).

5.2 HOST COMPUTER CAPABILITIES

Host computers in Pocnet provide the distributed processing nodes which will support individual POCCs and provide specialized nodes to promote resource sharing. Host computers may range in size from a dual-processor DBS system providing complex data handling capabilities to relatively small single-purpose minicomputers and perhaps even microprocessors. Most of the host computers in Pocnet will be components of model POCCs. Each model POCC will consist of one AP, one VIP, and one TAC; these model POCCs will handle the workload of GSFC control centers in the 1980s.

Each host will consist of a minicomputer of a size appropriate to the job that the host must handle together with the system software (operating system, language processors, etc.) which complement the hardware, making it a complete computing system on which application programs can be run. Each host computer will be connected to the other computing nodes in Pocnet and to the outside world through the Pocnet IPC system. Physically, a host will be connected to other Pocnet computers by a full-duplex ADCCP/HDLC serial line. To most computers, this line will appear to be an input/output device; hence, for each type of host computer, a device handler program must be written to monitor and control this link. Since on most hosts more than one program may wish to access IPC at the same time, some software must be provided to control the flow of data from programs running on the host into IPC and to sort out and buffer incoming messages.

On Pocnet, a running program is referred to as a "process" and the logical channels over which a process sends or receives data are called "ports" of the process. These

ports are simply the file names, data set names, or unit numbers used in input/output statements in applications programs. To take advantage of the resource sharing capabilities of Poccnet, a host must inform the network data operations control center (DOCC) of the names of all externally addressable process ports. This is accomplished by issuing DEFINE directives to the DOCC through IPC. When a process on the host wishes to establish a connection to a port of a process running on another host, the initiating process must issue a CONNECT directive to the DOCC, specifying the pair of ports to be connected. In some cases, a connection might be established by a process which is not itself part of the connection. For example, in Figure 5-1, the ground controller (GC) process running in the HEAO system in VIP5 sends to the DOCC through its output port (OUT) a directive to connect the input port (IN) of the telemetry handler (TMHDLR) of the HEAO system on applications processor 3 (AP3) to the output port (OUT) of the telemetry decommutation process (TLM) running on telemetry and command processor 1 (TAC1). To be able to establish the virtual channel between these two ports, the DOCC must know of their locations; thus, they must have been defined by the processes which own them before the CONNECT directive was issued.

The procedures to be followed in order to connect a host computer to Poccnet are described in the Poccnet Protocols Manual, Appendix 5A. Discussions of the following topics are included in this manual:

- a. Serial line electrical characteristics.
- b. Point-to-point physical communications, including access procedures, frame structure, and transparent procedures.
- c. Flow control in logical connections.
- d. End-to-end communication protocols, including message acknowledgment and error procedures.
- e. The host interface, including access procedures (connect, disconnect, define, error, and recovery procedures) and connection management.
- f. The user process interface to Poccnet.
- g. Process-to-process protocols, including data transfer protocols, use of virtual devices which are connected to VIPs, virtual user protocols, and the Systems Test and Operation Language (STOL).

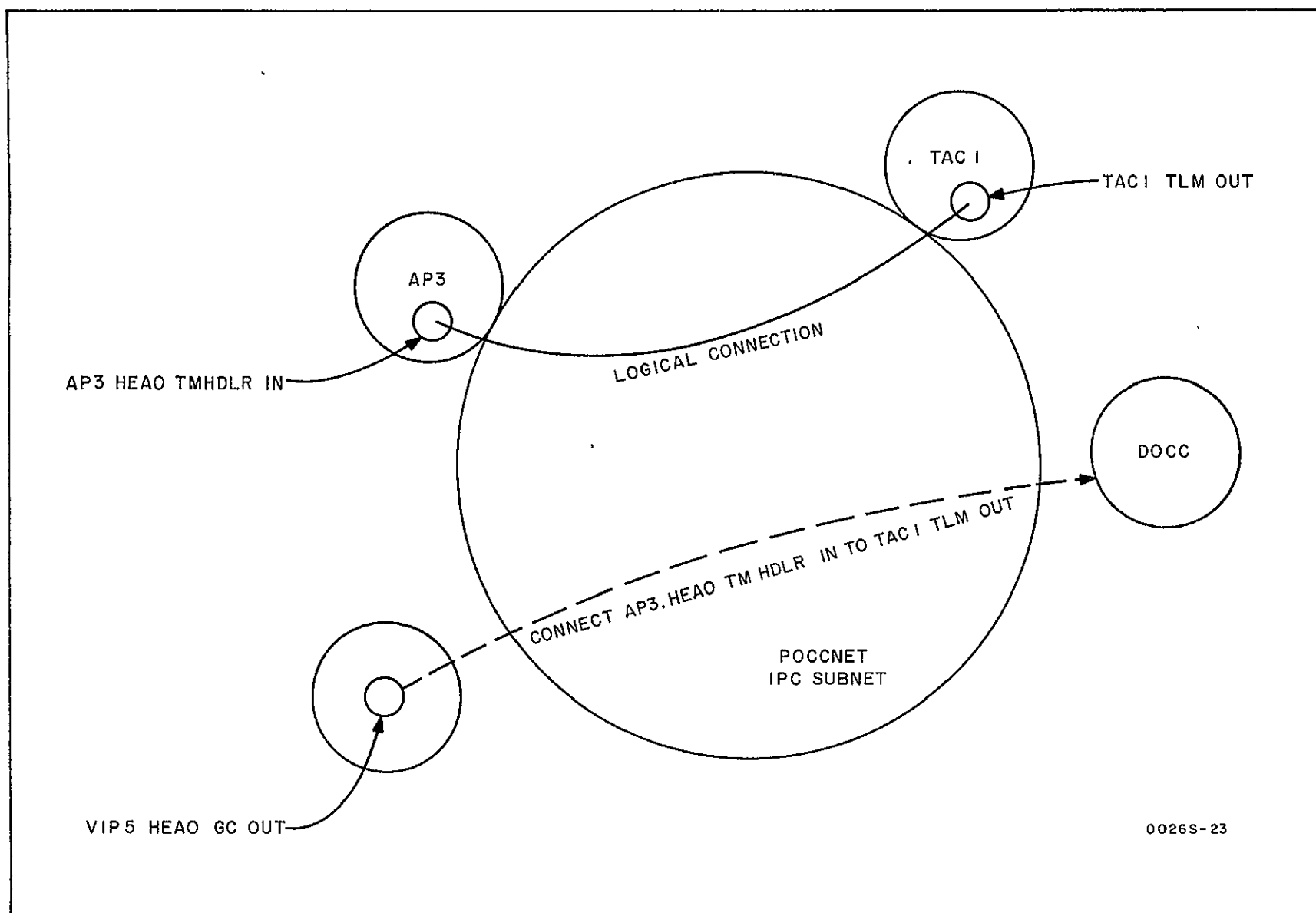


Figure 5-1. Establishing Logical Connections in Pocnet

The Pocnet Protocols Manual describes the resources and capabilities provided by Pocnet to help the POCCs do their work and sets forth the procedures to be followed in order to connect a host computer to Pocnet. A concise description of Pocnet's layered protocols is presented in Appendix 5D, A Layered Interface Structure for Distributed Systems Based on X.25.

5.3 APPLICATIONS PROCESSORS

Applications processors (AP) on Pocnet are general-purpose computing systems (computer plus operating system) capable of providing a process-oriented computing environment for applications and access to the Pocnet Inter-Process Communication (IPC) System. In addition, each AP supports the minimum level of Pocnet interoperability conventions, by which the hosts on Pocnet can be operated by a remote user.

APs are provided by individual systems within Pocnet. For example, in one proposed initial Pocnet system approach, MSOCC is expected to provide four APs (the XDS 930 replacements), HEAOCC provides two APs (also 930 replacements), the DBS system provides two APs, and the DOCC provides one. In addition, the MSC&AD PDP-11/70 real-time simulation system and the 360-65 will be interfaced as APs. The 360-65 will support simulation activities, command memory management, and on-board computers (OBC). Figure 5-2 shows these APs in the Pocnet environment. Each AP will, at any given time, be dedicated to support of one POCC or one Pocnet system. Thus, one AP will always support the DOCC function (described in paragraph 8.3), one or two will support the data base storage (DBS) system, and one (or more) will support each operational POCC. The APs together with the other Pocnet host computer systems and IPC make up a distributed computing system which will meet the requirements for low-cost, reusable, flexible ground support systems at GSFC in the 1980s. The philosophy of, and rationale for, this approach are documented in GSFC X-510-76-250, A Concept for Pocnet, and its appendix.

A Pocnet AP can be thought of as a computing mode or entity, a general-purpose virtual computing machine capable of taking a system load and running it, providing for all its process and file management, and building the system load in the first place.

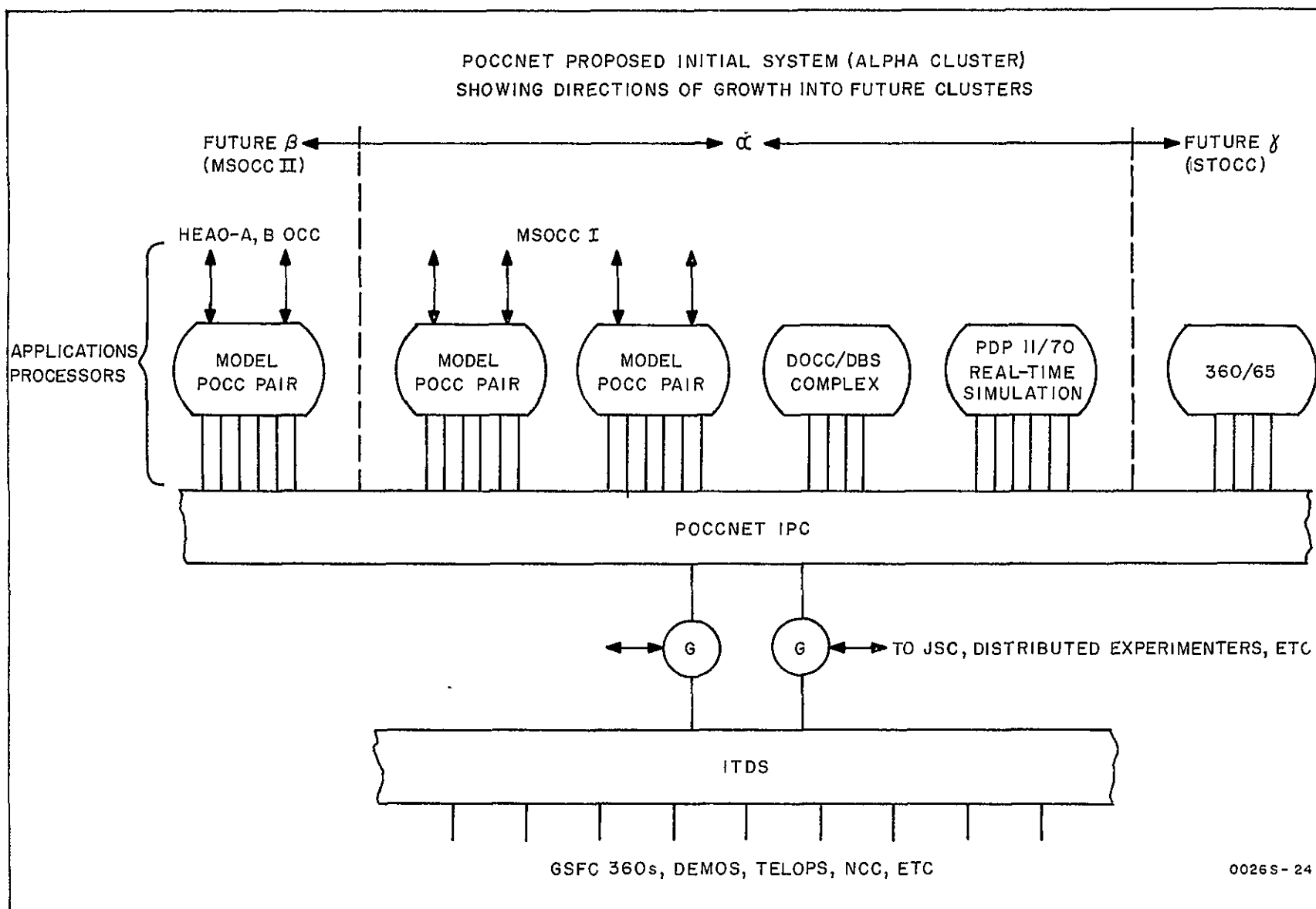


Figure 5-2. Applications Processors in Pocnet Environment

Each AP has the capacity of a PDP-11/70 and is capable of running all the applications programs in a modern POCC (such as AEOCC). APs are not special-purpose hardware but are general-purpose computers bought from a computer manufacturer together with a multiprogramming real-time operating system. Depending on how they are configured, APs could cost from \$100K to \$300K in 1977 and from \$10K to \$100K in 1985.

They are the computing nodes which "host" general computing pieces of a distributed computing system, and they are the AP subsystems of Poccnet.

The Poccnet use of APs is no different from POCC systems today at GSFC. Each POCC has one or more APs to run its applications programs. Basically, you buy as many APs as you have POCCs to run, plus a few more for backup.

The APs (PDP-11/70s running RSX-11M), which will replace the computers currently used in MSOCC, will support the model POCC applications described in paragraphs 3.3.2.2 and 7.2. The specifications for these APs, which were procured as XDS 930 replacements, are given in the manual Development and Integration of Special Hardware-Software Systems for the Multi-Satellite and HEAO Operations Control Centers, S-511-2, November 1976.

The DOCC AP will provide a center for control of Poccnet operations and POCC configuration. In this AP, applications programs will be run to schedule the use of other host computer systems and to allocate resources among the POCCs supported by Poccnet. The DOC will be able to remotely configure a POCC by establishing virtual connections between the AP which will actually run the POCC software systems and the other hosts required to support a given payload. These other hosts would normally include a telemetry and command system (TAC) and a virtual interface processor (VIP), along with connections to the DBS for data base service and operational checkpointing and to a gateway for access to support computing or outside communications. In the more sophisticated systems, the DOCC would be able to load the AP remotely and start POCC operations. It would also be able to monitor POCC operations to ensure proper functioning and could operate the POCC through the IPC.

The function of the DOCC is discussed in greater detail in paragraph 8.3. The DBS system is described in paragraph 5.4. The POCC APs will benefit by their connection to Poccnet because they will have access to a large quantity of on-line storage (on DBS)

and will have backup equipment available when needed for special operations or to substitute for malfunctioning units. Through the Poccnet IPC subnet, a POCC AP will be able to obtain services required to support its payload but which cannot be provided at reasonable cost by the AP itself. These services include interfaces to NASCOM, Telops, and support computing, as well as connections to outside communications facilities to provide for remote science operations centers.

5.4 DATA BASE STORAGE SYSTEM

5.4.1 INTRODUCTION

Decreasing hardware costs, increasing mission complexity and the availability of sophisticated data management software have resulted in increased use of data base techniques in control center implementations at GSFC in the last few years. This trend will accelerate in the future as standard, proven data base management systems become available on minicomputers and as system designers and programmers become aware of the advantages of the data base approach.

In a modern control center, the data base is the interface between the POCC computational system and the other elements of the ground support system, including experimenters, project planning personnel, external computing facilities, and data storage and cataloging facilities. The data base serves as the interface for data exchange among the various functional areas of the POCC.—The use of the data base for interfacing makes modifications or additions to the POCC computational system easier to design and implement and facilitates enforcement of security restrictions. The central control of data provided by the data base approach helps to maintain data integrity and encourages the design of software with a high degree of data independence. These qualities are prerequisites for the production of reliable and adaptable POCC computational systems.

In order for an AP to actually run an application program, it must be loaded initially with the programs and data that define that application system. In the past, these initial loads have been stored on magnetic tape, paper tape, cards, and instructions for the computer operator to enter data manually at a keyboard or front panel. The Poccnet orientation-to-people driver precludes the use of these for normal operations, since they require manual handling of storage media which is inefficient and unreliable.

On Poccnet, it is a principle of system philosophy that operations be "automatable". This implies that there exists the necessary systems mechanisms to automate any desired unit operation or sequence of operations. Whether a particular sequence of operations is in fact automated or manual will be a matter of policy not of mechanism. In order that systems operation be automatable, it is necessary that the system be loadable without operator assistance which precludes the use of storage media that requires manual handling. Hence, for automatability of initial system loading, a reliable on-line storage capability is required.

Additional requirements for continuously available on-line storage come from the need to periodically checkpoint records of each running system to allow for safe restarting from the last point of known correct operation following the occurrence of a fault event. In addition, a place to build up statistical records of operational systems is required for performance monitoring, evaluation, and tuning.

A large amount of highly reliable on-line storage is also required for storage and retrieval of data bases and program libraries. This storage would contain all important systems development information, such as spacecraft descriptions and tables, display and processing information, standard operational procedures, and software in various stages of development. This capability provides timely, accurate, and readily accessible information to both systems developers and systems users.

The DBS system responds to all these requirements for continuously available on-line storage. It provides standard file and data structure specifications, as well as file storage and retrieval services. There will also be provided a DBMS for use by personnel responsible for generating and maintaining payload and systems related data bases.

To make its services continuously available, the DBS subsystem could be implemented as a dual-processor system connected to all other host computers through the IPC subnet. A dual-redundancy system would provide a point of high reliability within Poccnet to provide backup or restart capability for the Data Operations Control Center (DOCC) system, the IPC PMPs, and the individual POCCs. Therefore, the dual-redundant design approach has been emphasized during the study phase and is reported upon herein. Nevertheless, most of the DBS requirements can be met with a designated uniprocessor system, and this is recommended for initial implementation.

Paragraph 5.4.2 presents the functional requirements for the DBS system. Subsequent sections present a functional overview, a proposed hardware configuration, a dual-processor coordination technique, backup and recovery procedures, an initial software implementation for the DBS subsystem, and a discussion of DBS applications software.

5.4.2 OBJECTIVES

The objectives of the initial implementation of the DBS system are as follows:

- a. Provide a data base definition facility and identify standard data and file formats compatible with VIP specifications.
- b. Provide a DBS hardware/software configuration which will ensure continuously available and reliable storage and retrieval of programs and data for POCCs and Pocnet subsystems.
- c. Provide a data base management system able to support POCCs and Pocnet subsystems non-real-time requirements for storage and manipulation of data bases.
- d. Provide an initial-load capability for hosts.
- e. Provide a checkpointing facility to ensure backup and restart capabilities for POCCs and Pocnet subsystems.
- f. Provide a service time for short requests (read a record, write a record) of 10 seconds or less. Service times for file transfer requests will depend on the amount of data to be transferred.
- g. Provide for automatic or semiautomatic initial loading of other hosts. Initial loading of small, core-only systems, such as PMPs or VIPs, will take 1 minute or less, while loading of APs may take 5 to 10 minutes, depending on system size and the IPC bandwidth available.

In addition to the specific objectives, a number of general objectives that the DBS subsystem should meet were identified during the Pocnet definition phase as follows:

- a. Minimize, to the extent possible, redundant data across POCCs and Pocnet subsystems.
- b. Minimize, to the extent possible, data inconsistencies.

- c. Maximize, to the extent possible, the sharing of data.
- d. Apply security restrictions.
- e. Maintain data integrity.
- f. Provide for data independence.
- g. Be expandable by allowing the addition of new storage with little or no impact.
- h. Be flexible by being able to support various storage-management techniques and by being able to support system reconfiguration with little or no impact.
- i. Be reliable by minimizing loss of data and by providing fail-soft and back-up procedures.
- j. Be cost effective by providing various levels of service as required.
- k. Be easy to use by providing a well defined and comprehensive human-to-machine interface.
- l. Be general purpose by providing, to the extent possible, all tools and interface mechanisms needed to support the data base storage and management requirements for all POCCs and Pocnet subsystems.

5.4.3 DBS FUNCTIONAL OVERVIEW

In the spectrum of data base design philosophies, there are two extremes: decentralized and centralized. In the decentralized case, the data base and its user are isolated from other data bases and users and any interaction between one configuration and another can, at best, be contrived with difficulty. In the centralized case, there exists a data bank or repository which contains data bases. Every user whose data base is in the repository interacts with it for any data base transaction he might wish to perform. The access mechanism for data is the same for each user, and interaction among users is highly simplified and often comes down to only a case of definition and permission.

Figure 5-3 graphically illustrates these two approaches. For simplicity, a user has been identified with his schema, Σ_n , which is his logical view of data, and the notation D/Σ_n can be interpreted as user n's view of the data repository. The Pocnet DBS

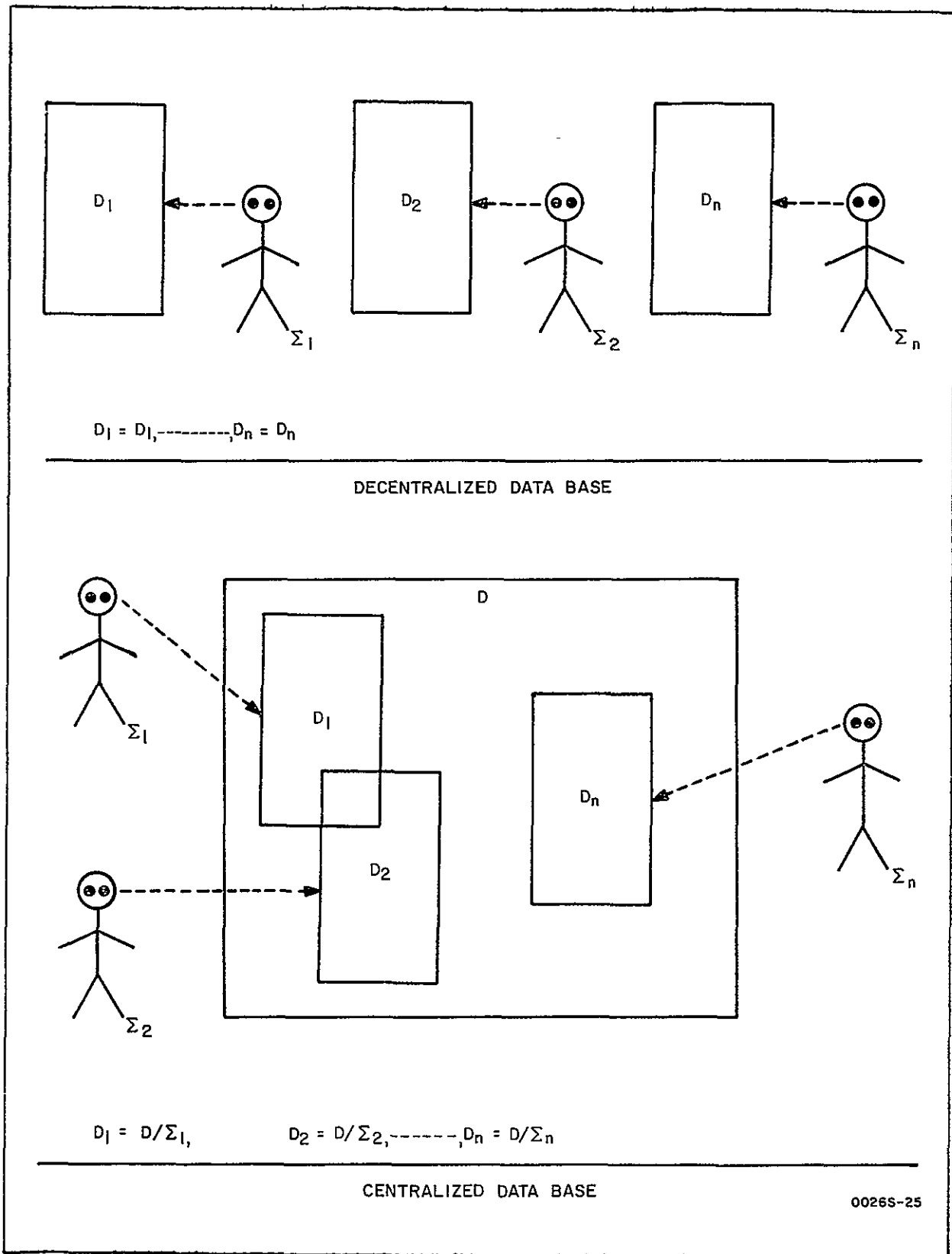


Figure 5-3. Centralized and Decentralized Data Base Philosophies

is a compromise between these two extremes. Each user in Pocnet must have access to his own logically and physically unique data base as well as access to data residing in the DBS repository. Every Pocnet user will make use of a common access mechanism to the repository, and, within user specified constraints, data in the repository may be shared among several users. Figure 5-4 depicts this DBS compromise.

In addition to the overriding objectives of high reliability and maintenance of continuously available storage, another consideration played a very important part in the evolution of the functional design of the DBS to be presented. It was felt that a system designed in the 1970s which will have its maximal use in the 1980s should take full advantage of proven state-of-the-art techniques currently available but be modularized to the point that subsequent incorporation of newer techniques, brought about in the dynamic and explosive area of data base management, could be realized with minimal impact to established user systems and in a cost-effective way.

One way to incorporate this consideration (and the one adopted in the proposed design) is to isolate the actual data base management system from the user with an intervening layer of interface which effectively makes the data base management system transparent to the user while still affording him all of its operational capabilities.

Figure 5-5 illustrates the proposed structure of the DBS. It is a two-level structure. Level 1, the user interface level, serves as the interface between the rest of Pocnet (the user) and the data base management level. Level 2, the data base management level, establishes the DBS interface with the actual physical repository of data. (A detailed discussion is contained in Appendix 5B.)

Level 1 is characterized by two functionally distinct activities, that is, processor coordination (for the dual-processor configuration) and translation. The mechanism of processor coordination in the context of the proposed physical configuration of the DBS (Figure 5-6) is exhaustively treated in Appendix 5B. The basic idea is that each user request is transmitted via the IPC to the DBS and is made known to each processor in the proposed configuration. It is the responsibility of processor coordination to determine which processor (assuming both are available) will respond to the request. Once this has been decided, the request goes to the translator on that processor. The translator essentially effects a mapping between the user request, which is stated in a high level user language, and the appropriate directive or set of directives for the data base management system.

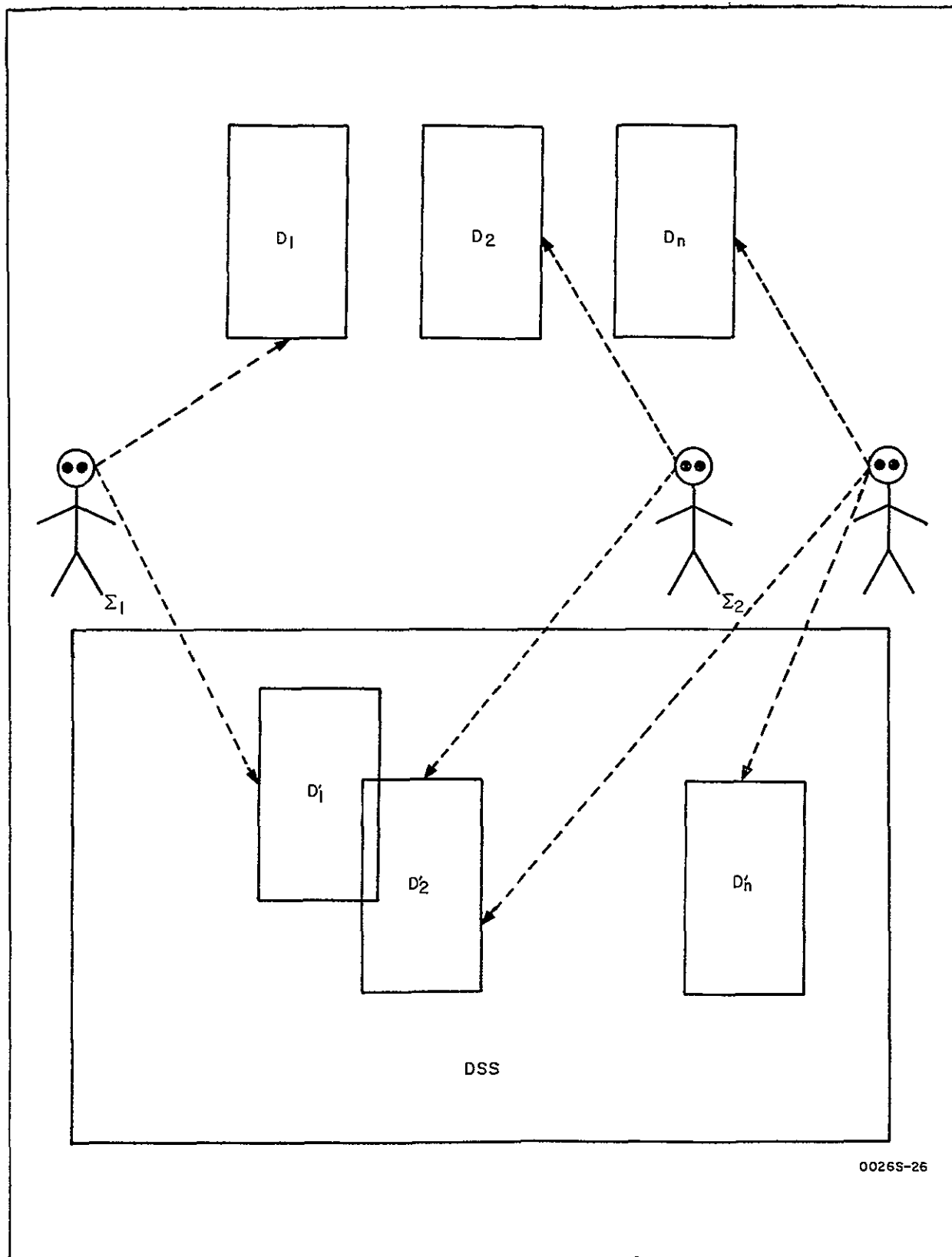


Figure 5-4. Pocnet Data Base Philosophy

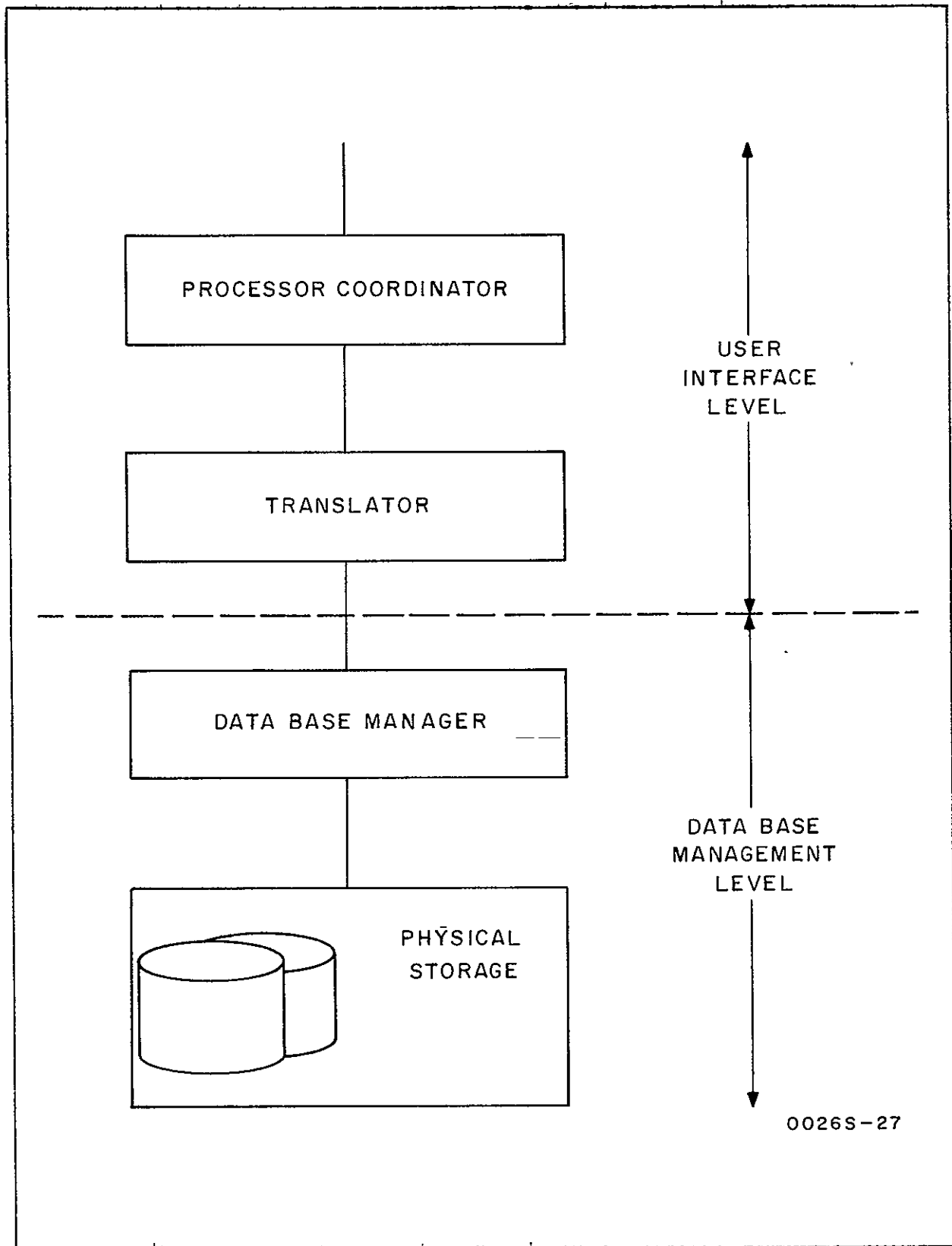


Figure 5-5. Proposed Structure of DBS

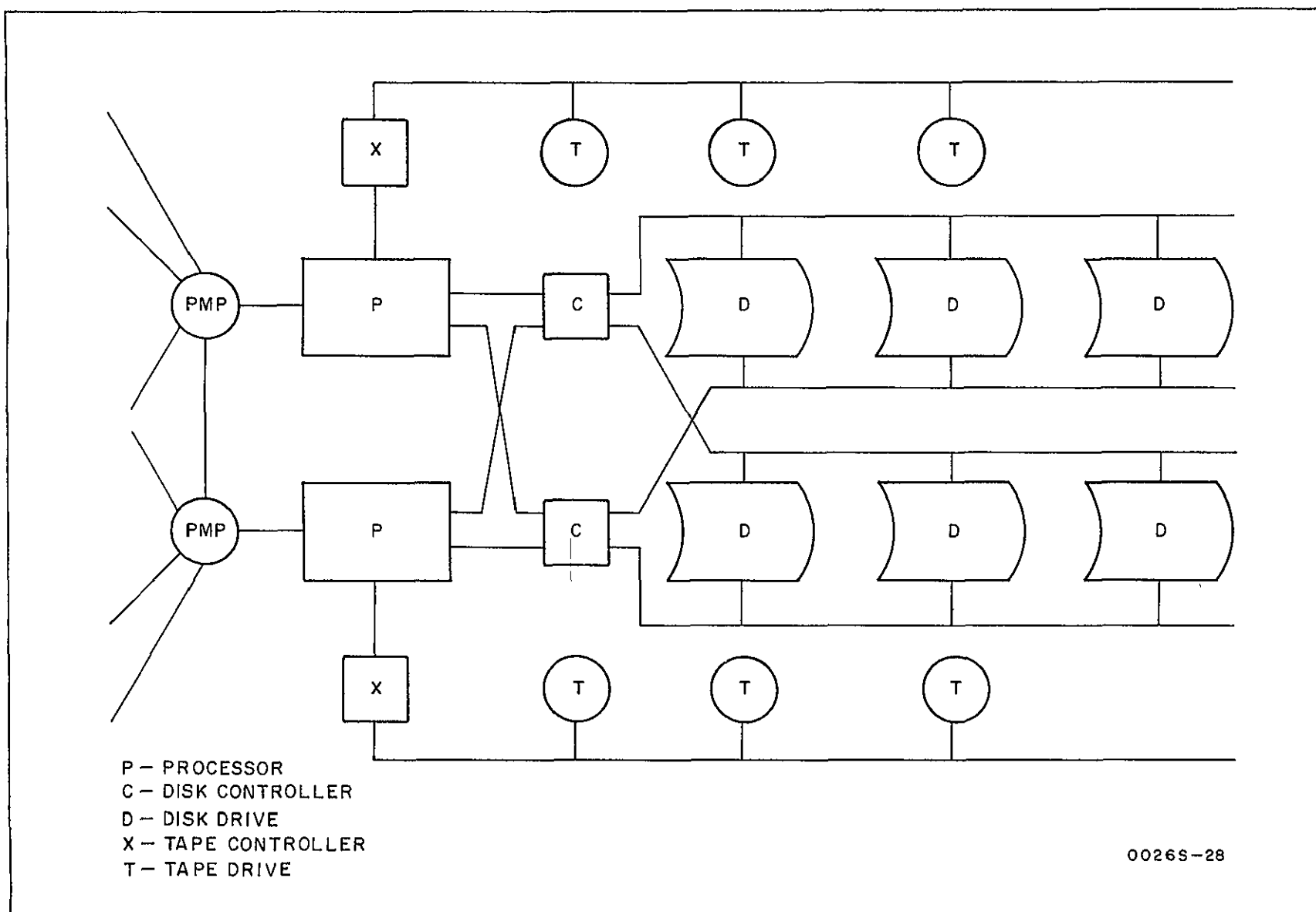


Figure 5-6. Pocnet DBS Subsystem Hardware Configuration

Level 2 consists of the actual data base management system and the physical repository. Functionally, the data base manager responds to the directive or set of directives produced by the translator by properly interfacing with the physical storage to carry out the user's request.

This technique allows DBS to change the data base manager, either by updating or replacing it, without affecting the user's previously coded application programs. A detailed discussion of this approach is presented in Appendix 5B.

5.4.4 DBS HARDWARE CONFIGURATION

The hardware configuration proposed for the Poccnet DBS subsystem is shown in Figure 5-6. The configuration is composed of two processors connected to tape drives and disk drives through their controllers. Each processor is connected to Poccnet through a separate PMP to ensure high reliability. Both processors are connected to each of the dual-port disk controllers and every disk drive is connected to two controllers. This configuration ensures that one of the two processors will be able to reach data on a given disk in the event of a processor or disk controller failure. The total on-line data storage capacity will be approximately two gigabytes (GB). Each processor will also be connected to a tape controller and several tape drives. These tape drives will be used to journal all data base transactions, to move data on or off line, and for data base backup/restore activities.

In developing this configuration, reliability analysis was used to compare alternative hardware configurations. By using information about the reliability of system components (CPUs, disk drives, etc.) quantitative measures were developed of the reliability of a system made up of these components. This reliability analysis enabled Poccnet designers to eliminate from consideration systems that even under optimum conditions would not meet the reliability requirements of the DBS. Reliability analysis also eliminated complex systems which provide little increase in reliability over simpler systems with lower cost. This work is described in detail in Appendix 5B.

The DBS subsystem will offer three levels of service, as shown in Table 5-1. All levels of service will feature virtually continuous availability of data storage and retrieval services and short access times to the primary on-line copy of a given file or data base. They differ in the access time to the backup copy. For level 1 data

Table 5-1
Levels of Service

Level	Primary Copy	Backup Copy	Primary Access Time	Backup Access Time
1	Disk	None	~ 10 sec.	∞
2	Disk	Tape	~ 10 sec.	~ 10 min.
3	Disk	Disk	~ 10 sec.	~ 10 sec.

there is no backup copy; therefore, if the primary copy is unavailable, access time is infinite. Level 1 service would therefore only be used for data which can be easily recreated. Typical uses would be for scratch files, print files, or other temporary data storage. Level 2 service provides a backup copy of data on-tape. When the primary copy (on disk) is unavailable, the backup copy can be accessed in a matter of minutes. The bulk of storage on the DBS subsystem would be committed to level 2 service.

Level 3 service provides an on-line backup, thus providing equally rapid data access from the backup copy as from the primary copy. All data and programs critical to real-time operation of Pocnet and individual POCCs would be maintained with level 3 service.

5.4.5 PROCESSOR COORDINATION

As discussed in paragraph 5.4.4, the DBS subsystem must use two CPUs to achieve the required level of reliability. This dual-processor configuration causes several problems in the scheduling of user requests for processing. We must ensure that each request sent from a user to the DBS subsystem through IPC will be processed by one of the two CPUs but not by both. We must also ensure that if one CPU fails the other CPU will take over processing of all outstanding user requests, at least to the extent of notifying the user that he must take corrective action. The procedures that will be used to resolve these problems are described in this paragraph. The two DBS subsystem CPUs will be identified as CPU A and CPU B, and the procedures will be discussed from the point of view of CPU A with the understanding that CPU B is governed by the same procedures.

The following assumptions were made in developing a solution to the processor co-ordination problem:

- a. Each request for service from a user has a unique name assigned outside the DBS subsystem (probably by IPC). This name could simply be the name of the process originating the request, with the date and time of the request appended.
- b. Each request is sent by IPC to one of the DBS subsystem CPUs, which then sends the request to the other CPU.
- c. A direct memory speed link exists between the two DBS subsystem CPUs.
- d. A teleprocessing monitor in each CPU receives and queues all messages coming into the DBS subsystem.
- e. A deterministic algorithm can be written to resolve conflicts between the CPUs over which is to process a given request. This algorithm might simply be that, in the event of a conflict, if the number representing the time of request origination is odd, CPU A will process the request, and if this number is even CPU B will process it.

A summary of the coordination scheduling process is presented here and is illustrated in Figure 5-7. (As mentioned before, the action will be described from the point of view of CPU A, but it is important to keep in mind that CPU B will be carrying out the same procedure in scheduling its work.) The steps involved for CPU A are as follows:

- a. Take a pending request from the input queue.
- b. Check with CPU B to make sure that it has not started processing this same request.
- c. Receive an "O.K. to process" message from CPU B.
- d. Send the request to the DBMS for processing.
- e. The DBMS services the request and sends a response to the user.

Note that in step c, when CPU B sends an "O.K. to process" message to CPU A, CPU B is explicitly agreeing not to process the request in question. If CPU B had already begun processing of the request when CPU A checked it, CPU B would have

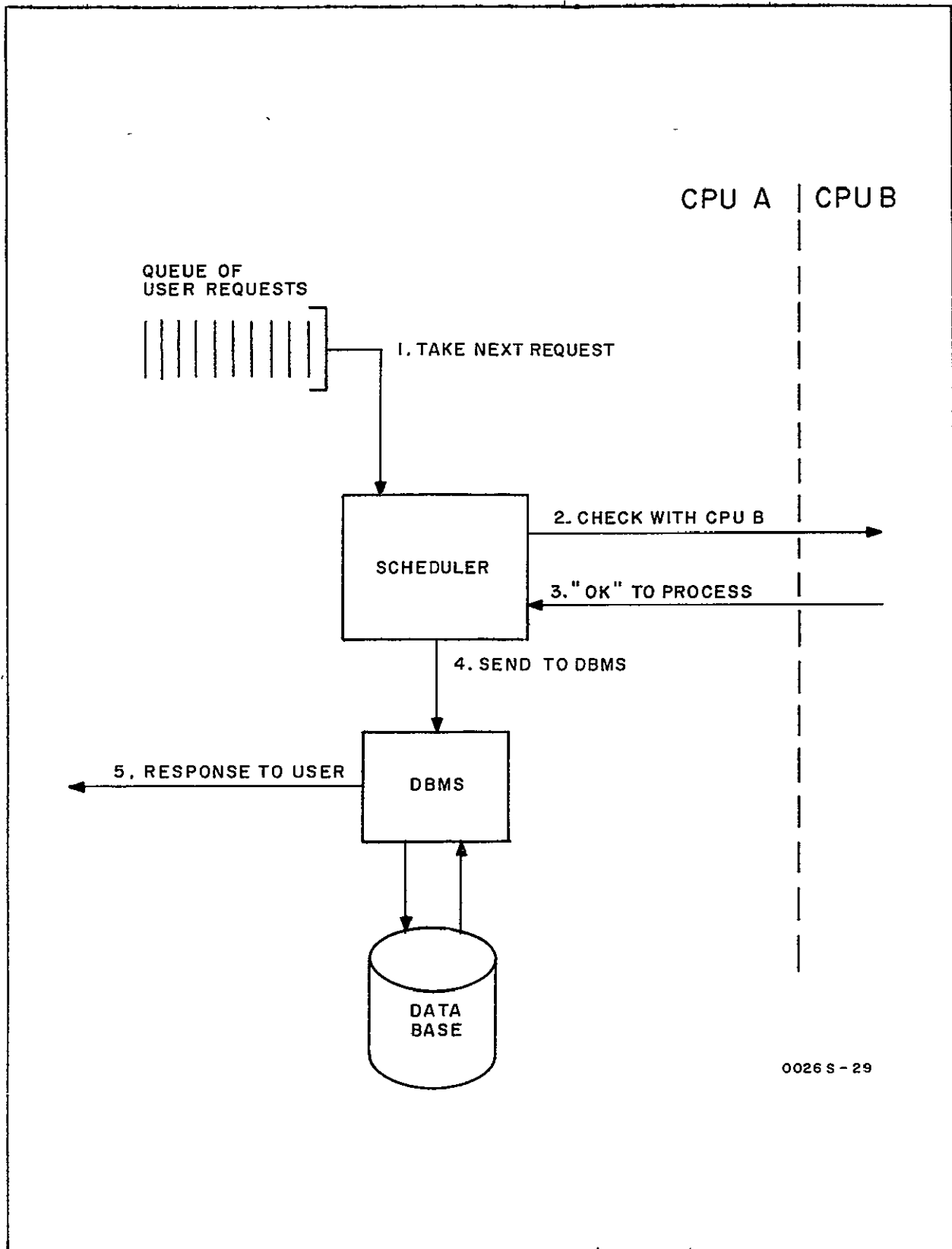


Figure 5-7. Overview of Scheduling

sent a "not O.K." message, causing CPU A to go back to step a, selecting another request for consideration. Thus, the two CPUs share the processing load by each taking a new request whenever it is ready to process another one. Two complications may arise from the simultaneous execution of this scheduling algorithm by both CPUs. First, if both processors should happen to choose (in step a) the same request for processing at the same time, it seems that neither CPU would give the other an "O.K. to process" message, resulting in deadlocking of the request. The second complication arises when one CPU fails; we must ensure that the other CPU picks up processing of all requests which were being handled by the CPU which is no longer operating. The detailed procedures described in Appendix 5B will correctly handle both of these situations. An additional benefit of the scheduling algorithm presented here is that it balances the workloads of the two DBS subsystem CPUs.

5.4.6 BACKUP AND RECOVERY

Since the data base management functions of the DBS subsystem will be provided by a commercial data base management system (DBMS), heavy use will be made of the vendor-supplied utilities which perform journaling, rollback/rollforward, and other data base backup and recovery procedures. We cannot, however, expect to find a commercially available DBMS which will handle the configuration switching problem that arises when one processor fails.

Briefly stated, the problem is that one of the DBS subsystem CPUs might fail at a time when it has one or more user requests in process. When this happens we must ensure that the remaining CPU picks up processing of all such requests.

To accomplish this, each CPU continuously runs two processes: a "heartbeat" process that periodically outputs a status message and a monitor process that receives the other processor's heartbeat message through the interprocessor link. Whenever one CPU fails and its heartbeat stops, the remaining CPU monitor detects this condition and initiates recovery. To effect recovery the monitor invokes a special recovery process which is described in Appendix 5B.

In paragraph 5.4.5, it was stated that CPU A checks with CPU B before initiating processing of a user request. It should be clear now that this step must be skipped if CPU B is not running. This can be accomplished by altering the scheduling algorithm, causing it to test the status word which indicates whether the DBS subsystem is running with one CPU or with two.

As mentioned in the introduction to this section, the DBS subsystem provides a point of high reliability in Poccnet which will facilitate backup of POCCs, and other Poccnet systems. This feature is especially important for the Data Operations Control Center (DOCC) host.

Both DBS CPUs will constantly monitor DOCC operations, and the DOCC itself will send checkpoint records to the DBS. In case of failure, one of the DBS CPUs will alert operations personnel and take over DOCC functions. Orderly restart will be accomplished using the latest checkpoint information. After any hardware or software problems have been corrected, the operator will be able to resume processing of DOCC functions on the original DOCC host computer, loading it from the DBS subsystem.

This backup system could provide triple hot redundancy for the DOCC. It is expected that a DBS processor will be able to take over DOCC functions in a very short time (1 to 5 seconds, estimated).

5.4.7 DBS SYSTEM SOFTWARE

This paragraph outlines the software implementation envisioned for the DBS subsystem. The following classes of software will be present in this system:

- a. A vendor-supplied operating system.
- b. A file management system.
- c. A purchased or leased DBMS.
- d. Poccnet-developed systems software.
- e. Software tools, purchased or GSFC developed.

The functions performed by each of these classes of software are described in the following paragraphs.

5.4.7.1 Vendor-Supplied Operating System. The vendor-supplied operating system will be used for process management and resource management. In the area of process management, the operating system will be responsible for interpretation of job control statements and process definition. It will control process execution including initiation, termination, and communication and synchronization among the various

DBS subsystem processes. The operating system will also handle most process input/output activities by providing access methods and device drivers. In the area of resource management, the operating system will be responsible for fault handling and reporting, memory management and peripheral allocation.

5.4.7.2 File Management System. A file management system will be used to provide the following functions: record and file definition, space allocation, directory maintenance, and control over use of files including privacy control. The file management system will either be provided by the computer manufacturer with the operating system or will be purchased from an independent software vendor.

5.4.7.3 Data Base Management System. A commercial DBMS will be obtained to provide a variety of important DBS subsystem services. The DBMS will provide a data definition language (DDL) which facilitates specification of data base structure and contents. It will provide a user-oriented language for data base access and for data base maintenance. It will provide access in several modes (such as interactive and batch) and will provide for shared access to data when required. The DBMS will interface with applications programs which generate source and object data base contents and will permit access to both source and object data bases.

In order to provide these services, the DBMS will perform space allocation and maintenance functions, journal all transactions, and perform backup/recovery procedures when necessary. The DBMS will control the flow of data into and out of data bases and will verify access rights before allowing use of data. The DBMS will also enforce data base access priorities.

The DBS subsystem must support source and object data bases for a number of payloads and for several Poccnet subsystems. Therefore, the DBMS used must be able to manage multiple independent data bases. If possible, the DBMS purchased or leased should support this feature; otherwise, extensive modifications may be necessary.

5.4.7.4 Poccnet-Developed Systems Software. A certain amount of systems software will be developed at GSFC for the DBS subsystem. This software will implement functions which are not performed by the commercial systems which will make up the bulk of DBS subsystem software. In general, the Poccnet-provided software

will be used to provide a better user interface than that available with commercial software and to increase commonality between DBS subsystem software and the other Poccnet subsystems.

More specifically, Poccnet must, if necessary, provide software to accomplish the following:

- a. To extend the file cataloging capabilities of the vendor-supplied file management system to permit the creation of named file hierarchies.
- b. To augment the DBMS to provide ultra-reliable, efficient journaling and backup/recovery procedures.
- c. To augment the DBMS space maintenance mechanisms, if necessary.
- d. To monitor DBS subsystem activity.
- e. To provide for the creation, maintenance, integrity, and use of ultra-reliable (presumably dual-copy) files.
- f. To augment the DBMS privacy control mechanisms.
- g. To augment the DBMS in enforcing priority schemes.
- h. To provide a data base control sublanguage.
- i. To provide a Poccnet-compatible language interface to the DBS subsystem languages.
- j. To interface the DBS subsystem operating system to Poccnet IPC.
- k. To control the dual-processor system.

5.4.7.5 Software Tools. A number of software tools will be purchased from commercial vendors or developed at GSFC to facilitate use of the DBS subsystem. These software tools will include the following:

- a. Utility programs to produce formatted listings of programs, data, file directories or data dictionaries.
- b. A report generator to create listings which selectively display data base contents.
- c. Compilers to generate object modules from source modules.

- d. A text editor to aid in preparation of source files and data templates.
- e. Utility programs to read or write entire files or sections of files.

5.5 VIRTUAL INTERFACE PROCESSORS

5.5.1 REQUIREMENTS

5.5.1.1 Need for Virtualization of I/O Devices. The desirability of virtualizing the low-speed computer peripheral devices used for payload support has been documented in A Concept for a Payload Operations Control Center Network (Poccnnet), GSFC X-510-76-250, November 1976. Briefly, the reasons for this approach are as follows:

- a. To facilitate the use of standard POCC applications software with devices from various manufacturers.
- b. To provide a standard, single, serial interface between the AP and the MOR.
- c. To reduce the difficulty of interfacing remote systems (such as science operations centers) to POCCs.
- d. To enable GSFC to upgrade I/O devices as new technology becomes available without major impact to existing software.
- e. To permit easy reconfiguration of existing facilities to meet changing mission requirements.
- f. To provide operational flexibility in the use of POCC facilities.

5.5.1.2 Number of Virtually Interfaced Devices Required. Missions planned or approved through 1983 (excluding Spacelab) will require up to 12 MORs. A total of 24 MORs will be used as a planning estimate for determination of the number of terminal peripheral devices required throughout the 1980s. The typical MOR in this period (based on observations of current POCCs and the new peripherals coming into the marketplace) is envisioned as having four to eight interactive display devices (color CRT/keyboards), one to two hard-copy devices (sociable printers), two to four eight-channel strip-chart recorders, and shared special-purpose devices (color graphics, voice synthesis/recognition equipment, etc.).

The estimated number of virtual terminal devices required for mission support in the 1980s is as follows:

Interactive displays	24 x 6 POCCs	=	144
Hard-copy devices	24 x 1.5 POCCs	=	36
Strip-chart recorders	24 x 1 POCC	=	24
Color graphics systems			6
Voice synthesizer/recognizers			<u>2</u>
Total number of terminals		=	212

5.5.2 FUNCTIONAL DEFINITION OF VIRTUAL TERMINALS

A virtual terminal (VT) device is a Poccnet process designed to control, or simulate, a physical input/output device and to make this physical or simulated device available to other Poccnet processes via IPC ports associated with the virtual device process. In this way, all control structures associated with input/output devices may be given standard forms; standard virtual device control messages are translated by the virtual device process into the specific forms required to control the physical devices by means of the hardware on which the process is executing.

Thus, a virtual device on Poccnet is an "ideal" concept of considerable generality. This most general device is known as a "random-access block display." Any implemented virtual device is a special instance of a RABD characterized by a subset of the attributes and capabilities of the RABD. —

The Poccnet Model POCC supports the devices contained in a single MOR on a distributed minicomputer called a virtual interface processor (VIP).

5.5.2.1 Random Access Block Display. The physical prototype of the RABD is a CRT display. The RABD has attributes and control characteristics which, however, considerably exceed those of existing real devices. In particular, the external physical medium is capable of being divided into nonoverlapping, named, rectangular subregions. Text outputting and inputting and the majority of VIP controls may be performed with reference to one of these regions. Associated with each region is also a system of protection.

The following three types of messages are interchanged between a process and a VT (through IPC). These are:

- Process-to-VT commands and data.
- VT-to-process format codes and data.
- VT-to-process sense responses.

All messages consist of combinations of literal data with additional constructs specifying the methods of interpreting literal data, such as device commands and modifiers. The commands are few in number (GET, PUT, MODESET, and SENSE) but are accompanied by parameters which qualify the basic actions in numerous ways.

5.5.2.2 Size System. Every VT is characterized by means of the dimensions of its physical input or output medium. In addition, the VIP (that is, the mainframe on which the VT process is executing) may make possible, through local storage, the maintenance of several "pages" (complete rectangular displays) of output data which can be randomly accessed at any time. The smallest addressable unit of the rectangular display is called a "pixel", and every alphanumeric (or other) graphic will be a rectangular matrix of pixels having certain dimensions constant for the VT. Moreover, each pixel may be displayed or read with at least two intensities (ON, OFF, and possibly other gray-scale or color values). The ensemble of values summarizing all of these physical or logical aspects of the rectangular display medium is called the size system. Explicitly, then, the size system consists of the following:

- a. (p_0, p_1) — The range of page-numbers employed ($p_0 \rightarrow p_1$).
- b. (r, c) — The numbers of row and columns of text displayable on one page.
- c. (m_1, m_2) — The number of pixels per row and per column encumbered by a single displayed text graphic (of the smallest size).
- d. (i_1, i_2) — The range of permissible intensity values that can be associated with each pixel ($i_1 \rightarrow i_2$).

Character positions within the rectangular display are enumerated (0, 0) (upper left-hand corner) through (r-1, c-1) (lower right-hand corner). As an example, the size system $((0, 3), (24, 80), (9, 7), (0, 7))$ characterizes the virtual RABD as consisting of four pages (numbered 0 through 3) of 24 rows of 80 characters; each character is a 9 by 7 rectangular matrix of pixels capable of eight intensity or color levels (0 to 7).

5.5.2.3 Notational Convention. The descriptions of device-level messages in the remainder of this section make use of a simple "source" or "publication" language (as distinct from the underlying bit patterns). Each command or device response will be represented as a series of keywords separated by commas. Each keyword may be postfixed with a parenthesized list of parameters and/or modifier keywords; the whole series of such constructs is terminated by a semicolon.

5.5.2.4 Text and Commands. Every message exchanged between a VT and a Poccnet process is composed of two kinds of components: literal text and commands. A text specification may take one of the following two forms:

- a. For output to VTs the form is TEXT (<literal string>). That is, text for output must be specified literally.
- b. For input from VTs the form is TEXT(n). That is, only the length of the (requested) text is specified.

Virtually interfaced terminals require explicit command elements, distinct from text, to specify the disposition of text. The basic command elements are as follows:

- a. Positioning verbs, such as PAGE, SKIP, and COLUMN TAB.
- b. Modifiers of the meaning of text, such as FRONT and COLOR.
- c. Parameters accompanying MODESET commands or returned in response to SENSE commands, such as ALLOCATE, READY, and COMMAND-REJECT.

The "publication form" of all the ancillary command elements is always a keyword, followed optionally by a parenthesized list consisting of keywords (for example, OFF), decimal constants, and character strings (which are not to be confused with text data).

5.5.2.5 Complete Commands. Each complete command from a Poccnet process to a VT consists of an operation code, plus zero or more text and modifier elements. The operation codes are as follows:

- a. PUT — Transmit literal text and/or virtual positioning information to the external medium.
- b. GET — Return text from the external medium and/or position the external medium.

- c. MODESET — Alter the internal state of the VT so that subsequent data and commands will be interpreted in a specific way.
- d. SENSE — Return data defining the readiness of the virtual device to accept further commands (SENSEREADY) or defining the current internal state of the VT (SENSEPARAMETERS).

After completing the actions implied by GET, PUT, or MODESET commands, a VT will automatically return to the Poccnet process the data requested by the SENSE-READY command; each GET, PUT, or MODESET command implies an additional SENSEREADY.

5.5.2.6 Positioning Controls. This discussion presents only the general character of medium positioning controls. The detailed enumeration of these controls and their effects is given in Appendix 5C.

Each VT, or named subregion of a VT, has an instantaneous state which is partially characterized by a pair of integer values (row, col), giving the position within the display region at which the next character to be displayed will be placed (PUT operations) or from which it will be retrieved (GET operations). This pair is called the CURSOR for the region. Its value is normally incremented for each additional character transmitted as follows:

- a. $col \leftarrow \text{mod}(col+1, c)$
- b. IF col = 0 THEN row $\leftarrow$ row+1

where the value of c in statement a is taken from the (r, c) parameter pair in the size system. The behavior of the cursor when statement b causes the condition row = r to occur depends on the nature of the physical devices supported by the VIP process as well as the MODESET commands previously issued to the VIP. For sequential devices (such as printers), the cursor will be reset to (0, 0) and a PAGE (forms-eject) operation will be forced. For randomly addressable devices (such as CRTs) the following two modes are available:

- a. In the PUSH mode, the contents of the currently displayed page are modified by shifting each row of text down one position; the bottom row is lost, row 0 is set to blanks, and the cursor is reset to (0, 0).

- b. In the FILL mode, new characters are transmitted until row = r-1; the contents of the current page are modified by shifting each row of text up one position; the top row is lost, row r-1 is set to blanks, and the cursor is set to (r-1, 0).

Detailed descriptions of the cursor positioning algorithms may be found in Appendix 5B.

However, it is possible to position the cursor in ways other than those implied by the default algorithm outlined above, namely, by means of the positioning commands PAGE, SKIP, COLUMN, POSITION (for RABDs), and TAB. The actions produced by these controls are described fully in Appendix 5A, the Poccnet Protocols Manual, Computer Sciences Corp., SCPB I-77-130, July 1977, and in Appendix 5C, Control of Virtually Interfaced Peripherals, Computer Sciences Corp., SCPB I-76-33, October 1976.

5.5.2.7 Elements Common to All VT Commands.

5.5.2.7.1 Region Names. In the case of VT devices having the character of RABD, it is possible to allocate and name rectangular subregions of the external medium. Each VT command addressed to such a subregion must include the NAME (< string >) parameter element.

5.5.2.7.2 Protection Keys. For each allocatable VT resource, there is a system of four levels of protection types as follows: GET (the lowest level), PUT-modify, MODESET, and lock-change (the highest level). Associated with each level is a "lock" string. Each attempted VT command must be accompanied by a password string and a level number. The attempted operation can be completed only when the password fits the lock of the indicated level and the level number is high enough for the type of command. Details of this scheme are presented in Appendix 5A.

5.5.2.8 MODESET Commands. The following MODESET functional capabilities are provided in the VT design as follows:

- a. ALLOCATE — Allocates a named subregion of specified size.
- b. BLOCKDISPLAY — Establishes block-display mode for a RABD.
- c. CHECKDEFAULT — Specifies a default substitute for erroneously transmitted data characters.

- d. COLORDEF — Defines a mapping between source and object color codes.
- e. DELIMITER — Specifies the end-of-page/file delimiter string.
- f. DISPMODE — Establishes FILL or PUSH mode for RABDs.
- g. FONTDEF — Defines the correlation between external and internal character representations for a specified font.
- h. FREE — Frees an allocated subregion of a RABD.
- i. FTABS — Specifies and enables/disables field-tab positions.
- j. LOCKS — Establishes lock strings for VT protection.
- k. STRING — Places RABD into in-string/out-string mode.

5.5.2.9 SENSE Commands. The following two categories of SENSE commands provide a Poccnet process with information about a VT device:

- a. Information about the characteristics and internal state of the device (the size system, tab settings, etc.).
- b. Information about the readiness of the device to accept further commands or about the success of the previous command.

Accordingly, and since the device characteristics form a rather large body of data, two types of SENSE commands are provided in VT design: SENSEReady and SENSE-PARAMETERS. The possible responses to SENSEReady are as follows:

- a. READY — The device is operational and ready to execute VT commands.
- b. BUSY — The device is currently executing a previously issued VT command.
- c. ENDMEDIUM — The physical input/output medium has been exhausted.
- d. ENDFILE — A GET command has caused the VT to encounter the end-of-data delimiter string or a physical end-of-file mark.
- e. OPERATOR REQUIRED — The device cannot be operated until an operator intervenes (forms jam, etc.).
- f. UNAVAILABLE — Operator must ready device or an ALLOCATE modeset command cannot be executed.
- g. RESERVED — Another process has OPENed the device for exclusive access.

- h. NOTOPERATIONAL — The device requires maintenance.
- i. DATACHECK — Irrecoverable data transmission error has occurred.
- j. COMMANDREJECT — A command has been decoded which cannot be given an interpretation for the device (for example, a COLORDEF modeset for a card reader).
- k. ATTENTION — The user has raised the attention condition.

The following information is returned for the SENSEPARAMETERS command:

- a. SIZE — The parameters of the size system for the device or named region.
- b. FTABS — The currently defined field-tab positions.
- c. FONTS — A list of identifiers, each specifying an available font definition.

The detailed listing and explanation of the data returned for SENSE commands is presented in Appendix 5A.

5.5.2.10 PUT Commands. A PUT operation consists of TEXT segments and controls completely separated; no text character is to be interpreted as having a control function. The following modifier elements are provided for PUT commands:

- a. LOCATION — Set cursor to specified row and column position (RABD).
- b. PAGE — Top-of-form operation. _____
- c. SKIP — Skip specified number of output or input logical records.
- d. COLUMN — Move cursor to indicated column.
- e. FSPACE — Forward-space the specified number of input/output records.
- f. BSPACE — Backward-space the specified number of records.
- g. HTAB — Horizontal tabulation.
- h. VTAB — Vertical tabulation.
- i. FTAB — Field tabulation (RABD).
- j. EMPHASIS — Apply specified emphasis characteristics to current field.
- k. FONT — Use specified font.
- l. TEXT — Transmit specified literal text.

Details of the meaning and use of these command elements will be found in Appendixes 5A and 5C.

5.5.2.11 GET Commands. A GET command is analogous to a PUT operation in many respects; controls have cursor-modification functions like those of PUT, but no text is transmitted in the command itself. A field width, but no specific data, is associated with the TEXT verb. Modifier elements for GET operations are generally similar to those for PUT. The following modifiers have meanings differing from those for PUT:

- a. DECK — Move to the input record immediately following the next delimiter or end-of-file mark (analogous to PAGE).
- b. FONT — Interpret incoming data as belonging to the specified coding scheme.
- c. TEXT — Transmit the specified number of characters beginning at the current cursor position.
- d. MODIFIED — Transmit only modified fields.

Detailed specification of the effects of these command elements will be found in Appendix 5C.

5.5.2.12 Object Coding of VIP Commands. A simple scheme for representing the elements of VIP commands as compact byte strings has been developed; the encoding rules are, however, not presented in this document. The reader is referred to the appropriate section of Appendix 5C.

5.5.3 VIRTUAL USERS — FUNCTIONAL DEFINITION

This paragraph describes a message transaction scenario which, in abstract terms, occurs frequently in interactive contexts; this scenario is formalized and a special Poccnet process, called a virtual user, is defined. The basic structure of the scenario is shown in Figure 5-8.

A Poccnet process (here called JOB) requires dynamic feedback from a controller process (here called VU). The information required to effect this feedback is transmitted through three IPC channels to/from three ports (PROMPT, ERROR, and RESPONSE) belonging to VU. The basic scenario is as follows:

- a. Step 1 — JOB requests control information from VU by sending a prompt message to the port VU.PROMPT, then waiting.

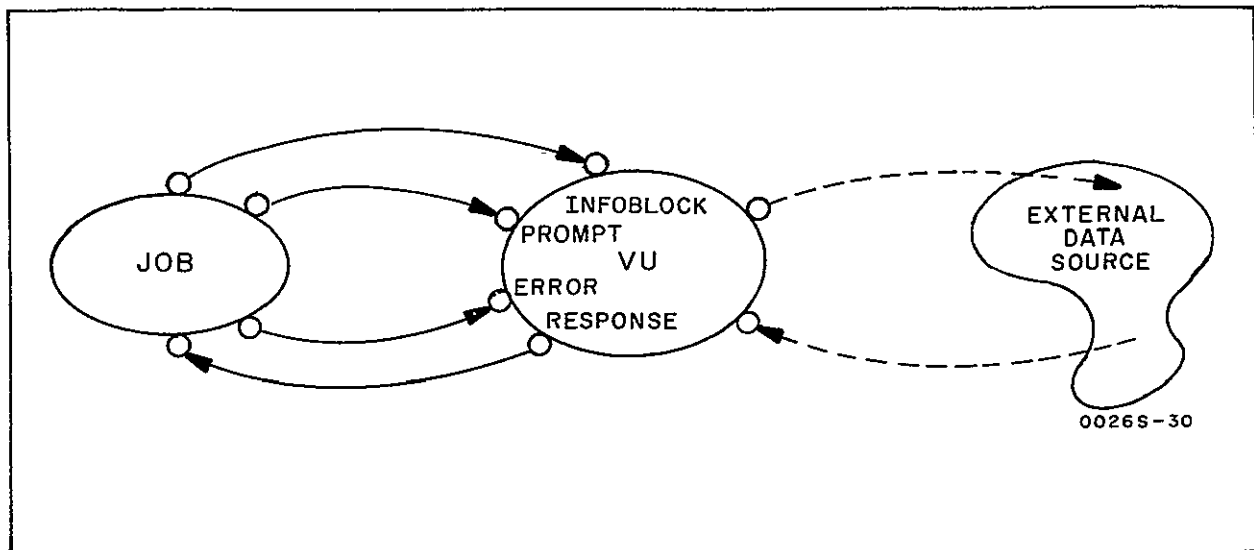


Figure 5-8. Virtual User Scenario

- b. Step 2 — VU transmits the response back to JOB via VU.RESPONSE and waits.
- c. Step 3 — JOB either accepts this response or adjudges it to be erroneous in some sense. If the response is accepted, the scenario continues with step 1; otherwise, the scenario goes to step 4.
- d. Step 4 — JOB provides a feedback message to VU via VU.ERROR, then waits. VU formulates a revised response and the scenario continues with step 2.

In addition, JOB may transmit data at any time to the VU.INFOBLOCK port; this occasions no direct response from VU.

When the process VU (with ports ERROR, PROMPT, RESPONSE, and INFOBLOCK) is connected in this way to any other Poccnet process and obeys the protocols implied by steps 1 through 4, it is called a virtual user. This definition presupposes no internal structure or mechanism for the process VU nor does it exclude the possibility that VU has additional I/O ports beyond those connected to JOB (in the example above). Thus, VU may respond in one of the following ways:

- a. Compute all responses to PROMPT and ERROR messages from JOB.
- b. Use external storage to retrieve cataloged responses.
- c. Interact with a human user (that is, perform the function of a data prompter/editor).

- d. Obtain information from other Poccnet processes.
- e. Employ any combination of information sources given in steps a through d.

The interactive character of a virtual user implies that, at most, one RESPONSE or PROMPT/ERROR message is being processed at any time.

5.5.4 HOST REQUIREMENTS FOR VIP AND GATEWAY PROCESSORS

The mainframes hosting the VT and gateway (refer to paragraph 5.5) processes (V/G hosts) are concerned principally with the inputting, outputting, analysis, and transformation of coded strings of bits or characters; that is, they deal predominantly with symbolic rather than inherently arithmetic data. V/G hosts are merely links between a Poccnet process and another remote process or I/O device. Consequently, speed must be emphasized in order to prevent V/G processors from becoming bottlenecks in the system as a whole. Moreover, the individual processes executing in V/G hosts must be regarded as resources shared by an indefinite number of other Poccnet and remote processes. Thus, their implementation must rely heavily upon reentrant coding. The implications of this for both mainframe and operating system architectures are discussed in the following paragraphs.

5.5.4.1 Multiprogramming and Reentry. The fact that V/G hosts serve multiple simultaneous users implies that they should support a dynamically variable degree of multiprogramming. Accordingly, the vendor-supplied (or user-written) operating system should provide a method of creating tasks or subtasks and of monitoring them. Since all dynamically created subtasks are merely copies of a small number of basic (probably resident) programs, it is important that a task (or a subtask, etc.) be definable as a triple:

Task = (invariant reentrant executable code, task control block, working storage for current invocation).

In order that an executable code embodying a particular function need not be duplicated for each task invocation, it should be possible to write a reentrant code making use of base-plus-displacement addressing so that only the base address of the task working storage (in a CPU register) need be supplied at task-creation time. This also places a premium on a large number of general-purpose CPU registers and efficient register-to-register operations.

5.5.4.2 String Handling. The string-manipulating nature of V/G hosts makes certain kinds of nonarithmetic instruction highly valuable. These include such string operations as move, compare, translate, and inspect/verify.

5.5.4.3 Control Blocks, Buffers, and Working Storage. The coding of V/G processes is heavily dependent upon operations such as allocating and freeing blocks of processor storage used as buffers, control blocks, and working storage.

5.5.4.4 Microprogramming. The paucity of instructions for string handling and list processing in most minicomputer mainframes suggests that vendor- or contractor-supplied microprogramming be used to implement efficiently heavily used operations of these types.

5.5.4.5 I/O Channels. Since V/G hosts are essentially elaborate I/O control devices, it is important that they possess numerous cycle-stealing data channels and that highly multiported processor storage be available to minimize the contention for memory cycles.

5.5.4.6 Timing. Both the CPU and the operating system should provide facilities for maintaining time-of-day in milliseconds and for awaiting certain specified timing intervals. This is especially true for purposes of task-to-task protocols involving remote processes.

5.6 GATEWAYS

5.6.1 GENERAL CHARACTERISTICS

Gateways provide for process-to-process communication across network boundaries. Several forms of this communication can be identified as follows:

- a. Gateway-to-modem.
- b. Gateway-to-local CPU adapter.
- c. Gateway-to-local gateway.

Gateway to modem is the most general, since it permits the transfer of data to/from any remote node: computer, user terminal, RJE terminal, data acquisition hardware,

communications controller, etc. However, this form of communication is limited in speed to teleprocessing rates (usually less than 100 kbps, and often much less) and may be error-prone owing to the use of voice-grade signal carrier facilities. This form (or acoustic coupling) is expected to be used for supporting remote system development by programmers.

Gateway-to-local-CPU-adapter is the most efficient since it permits very high-speed transfer of data between a local data processing system and a Poccnet process (through IPC). However, it requires special, rather expensive equipment and custom software, for each type of local processor requiring a data connection to Poccnet. This form is expected to be used for interconnecting Poccnet to the M&DOD 360s via ITDS.

Gateway-to-local-gateway lies between the first and second types in regard to both speed and generality. An example would be communication between a gateway and a local Arpanet IMP. This form is expected to be used for supporting science operations centers located at universities.

5.6.2 FUNCTIONAL DESCRIPTION OF GATEWAYS

In transferring information into and out of Poccnet, the gateway processor must effect a translation between the protocols of Poccnet (at the relevant levels) and the protocols adhered to in the environment of the remote system. These protocols, which vary considerably from one network or remote processor to another, are intended to monitor and control both the validity and the flow of data. It is evident that an exact functional description cannot be given for gateways in general, since this description would have to hinge upon the characteristics of the remote system in question. Moreover, gateways will be expected to establish connections between Poccnet processes and remote processes; the method of doing this depends both on whether the initiator of a connection resides in Poccnet or is remote and on the method (if any) of invoking a process in the remote facility. Similar remarks hold for the problem of cleanly breaking such a connection. As indicated in Figure 5-9, gateways control the following three kinds of communication for each pair of processes connected:

- a. Data exchanged between the processes.
- b. Remote-to-gateway protocols.
- c. All Poccnet protocols up to the process-to-process level.

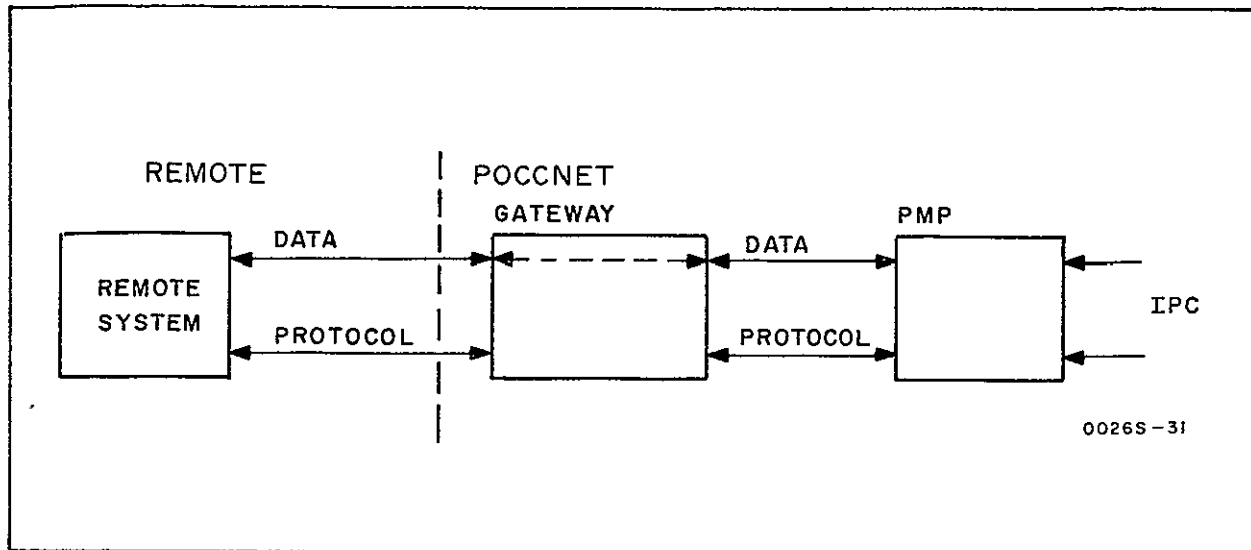


Figure 5-9. Gateway Data Paths

That is, the gateway makes the remote system seem like a Poccnet process and makes the Poccnet process seem like a remote terminal (of some sort) to the remote process.

5.6.3 INTERFACES TO BE PROVIDED INITIALLY

This paragraph enumerates three gateway applications to be provided in the initial Poccnet operational environment. For the reasons advanced in paragraph 5.6.2, it is not possible at this point to be completely precise about these interface functions.

5.6.3.1 Telops. The interface with the Telops will be through the output processor; at this time, all that is known about the interface is that some standard teleprocessing protocol (for example, asynchronous or binary synchronous, probably point-to-point half-duplex) will be used. Once the exact character of the communications protocol of the output processor system is known, the functional description of the gateway software can be given in full.

5.6.3.2 Johnson Space Center. At this time, even less is known about the nature of systems at JSC (to which interface will be required) than is known about Telops. Perhaps such communication would be ADCCP and X.25 via Telenet at a rate of 9600 baud.

5.6.3.3 System 360 at Goddard Space Flight Center. The interface to the System 360s at GSFC will be made by means of the Network Systems Corporation network adapter and ITDS. This device behaves as a control unit from the point of view of all computer systems attached to it via data channels. Consequently, although it is not presently known how the System 360 support of this device will be designed, it may at least be stated that the protocols involved will be quite simple, since the gateway processor will appear to the System 360 as a device similar to a disk controller, tape controller, etc. Therefore, the gateway will simply act as the terminus of Poccnet protocols, with virtually no requirement for Poccnet-to-System-360 protocol transformations.

SECTION 6
INTER-PROCESS COMMUNICATIONS

SECTION 6

INTER-PROCESS COMMUNICATIONS

CONTENTS

<u>Paragraph</u>		<u>Page</u>
6.1	Requirements Analysis	6-1
6.1.1	Functional, Operational, and Performance Requirements	6-1
6.2	Design Alternatives	6-2
6.2.1	Common Access Memory	6-2
6.2.2	Crossbar Switching	6-3
6.2.3	Common Bus	6-4
6.2.4	Cable Bus	6-5
6.2.5	Basic Loop Design	6-5
6.2.6	Computer Subnet	6-6
6.2.7	Conclusions	6-7
6.3	Operational Description	6-9
6.3.1	Overview	6-9
6.3.2	General Description of PMPNET Access Protocol .	6-11
6.4	Pocnet Message Processor Description	6-11
6.4.1	PMP Characteristics	6-11
6.4.2	PMP Software Organization	6-12
6.4.3	Frame Queue	6-12
6.4.4	Message Routing Table	6-14
6.4.5	PMP Statistics	6-17
6.4.5.1	Traffic by Physical Channel	6-17
6.4.5.2	Traffic by LCI	6-18
6.4.6	PMP Timing Estimate	6-20
6.4.7	PMP Core Estimate	6-22
6.5	IPC DOCC Description	6-22
6.6	Establishment of a Minimum Pocnet System	6-23
6.7	User-Level Commands	6-24

ILLUSTRATIONS

<u>Figure</u>		<u>Page</u>
6-1	Common Access Memory	6-3
6-2	Crossbar Design	6-3
6-3	Common Bus Design	6-4
6-4	Loop Design	6-6
6-5	Computer Subnet	6-7
6-6	AP and IPC Subsystem	6-9
6-7	Process-to-Process Communication Sequence	6-10
6-8	End-to-End/Point-to-Point Transmission	6-10
6-9	Point-to-Point Acknowledgement	6-11
6-10	PMP Software Organization	6-13
6-11	Frame Queue in the PMP	6-14
6-12	Source, Network, and Destination LCIs	6-15
6-13	Example of Message Routing Tables	6-16
6-14	Sample Traffic-by-Physical-Channel Table	6-17
6-15	Sample Traffic-by-LCI Table	6-19
6-16	IPC/HOST DOCC	6-23

TABLES

<u>Table</u>		<u>Page</u>
6-1	Requirements and Design Alternatives	6-8
6-2	CPU Degradation	6-20

SECTION 6

INTER-PROCESS COMMUNICATIONS

6.1 REQUIREMENTS ANALYSIS

There is a continuing trend within the POCCs to perform processing requirements by using low-cost high-performance minicomputers. For example, a PCM telemetry simulator can be developed by equipping a minicomputer with a few special features such as a shift output register and some software, or a PCM decommutator can be developed by equipping a minicomputer with a programmable pattern recognizer and some status lights.

The success of this approach as an answer to the special-purpose needs of the POCC is quite evident by taking a survey of the existing control center computer population. Virtually every control center has one or more minicomputers doing special tasks. Included are minicomputers from Honeywell, Digital Equipment, Interdata, and Varian. Unfortunately, the low cost of the minicomputers makes it unlikely that they will be surrounded by a complete enough set of peripheral equipment to make software modification convenient. Somehow, they must be connected to larger computer systems so that the peripheral resources can be shared by the impoverished minicomputers.

A mechanism for resource sharing is needed to make the most cost-effective use of a complex of many computer systems. Each computer could have a channel connecting it to this computer communication network for resource sharing or any other computer-to-computer communication need.

6.1.1 FUNCTIONAL, OPERATIONAL, AND PERFORMANCE REQUIREMENTS

This section will list only the requirements (functional, operational, and performance) for the Inter-Process Communication (IPC) network. A detailed discussion of these requirements is presented in Appendix 6A.

The functional requirements list includes the following:

- a. Flow control, error control, and message characteristics.
- b. Real-time host communication.
- c. Expandability.
- d. Manufacturer independence.

- e. Serial communication.
- f. Communication reliability.
- g. Geographical independence.

The operational requirements list includes the following:

- a. Easy to manage.
- b. Easy to use.
- c. Fail soft.

The performance requirements list includes the following:

- a. Channel speed.
- b. Message length.
- c. Throughput time.
- d. Response time.

6.2 DESIGN ALTERNATIVES

This section describes IPC design alternatives. Information from Anderson and Jensen¹ has been used concerning the identification of IPC designs and a common context in which to discuss them. Only general designs are documented in this section. Paragraphs 6.3 through 6.7 discuss the details of the recommended design (the computer subnet approach).

6.2.1 COMMON ACCESS MEMORY

In the common-access-memory approach (Figure 6-1), two or more hosts communicate through the common memory.

Hosts can be arbitrarily added and the in-transit message capacity can be increased by simply increasing the size of the memory. The logical complexity of the design is also quite low.

However, if each host is provided a direct channel, the number of available memory ports to the memory could be exceeded by an additional host. If the memory is

¹G. Anderson and E. Jensen, "Computer Interconnection Structures: Taxonomy, Characteristics and Examples", Computing Surveys, Volume 7, No. 4, December 1975.

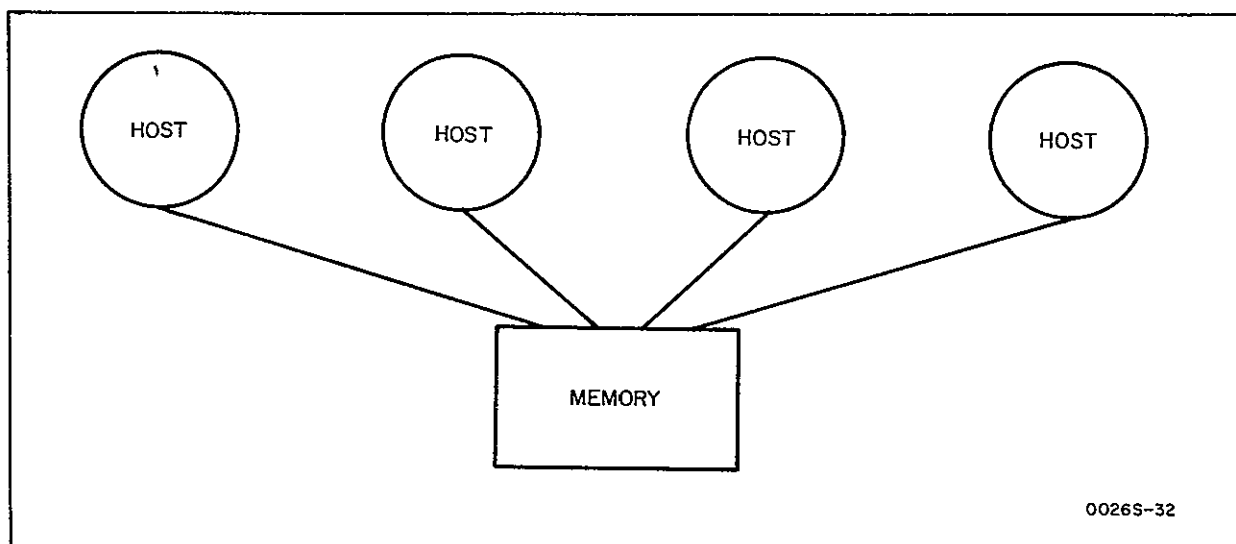


Figure 6-1. Common Access Memory

accessed through a shared bus, the memory bandwidth becomes a restriction on communication rates. The failure-effect characteristics are poor in the event of a failure of the memory unit or of the shared bus. Faulty software can also influence the failure-effect characteristics since software in the hosts normally has unrestricted access to the common memory.

6.2.2 CROSSBAR SWITCHING

Crossbar switching can be used to interconnect the different hosts. Figure 6-2 illustrates one such design developed for the Poccnet study. A DOCC computer controls the switch settings of the crossbar. (Appendix 6B documents the details of this approach.)

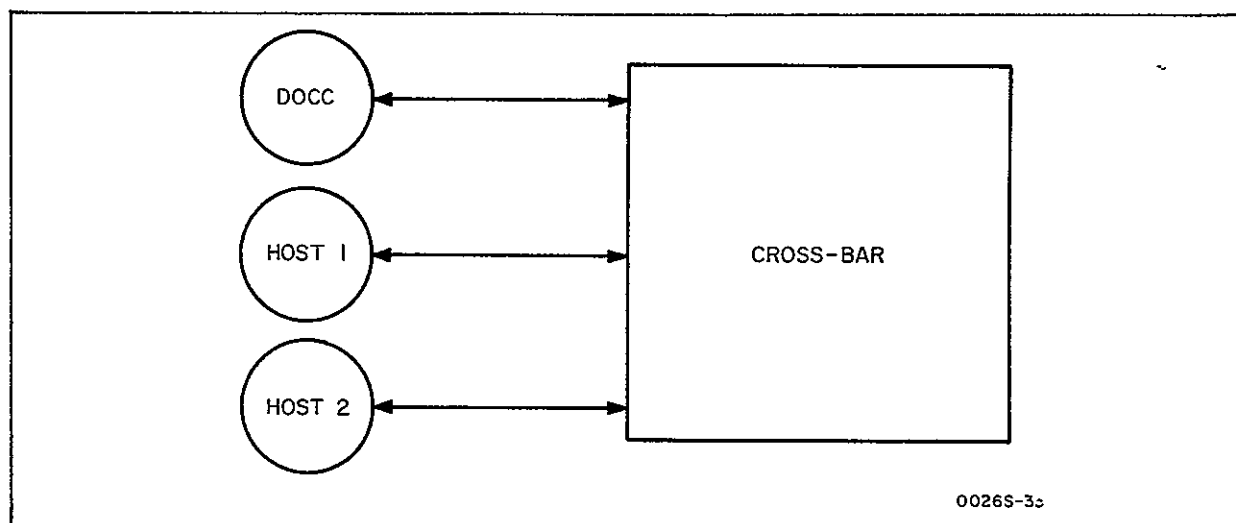


Figure 6-2. Crossbar Design

Upon a signal from the DOCC computer, each host transfers control information to the DOCC through the switch itself. Based on this control information, the DOCC would then configure the crossbar for the message transfer phase between the hosts. Hosts can readily be connected to the crossbar with little effect on the other hosts and the crossbar can also easily be enlarged.

A DOCC failure will make the system inoperative or locked in the last crossbar setting. A hot standby DOCC computer could improve reliability. For purposes of synchronization, the message cycle time of the switch must be fixed for the longest possible message length. If the message lengths are less, the throughput capability for a specific channel is reduced. As the number of hosts in the system increases, the cost of a crossbar switch becomes prohibitive.

6.2.3 COMMON BUS

A number of hosts are interconnected by a common bus (Figure 6-3) in the common bus approach.

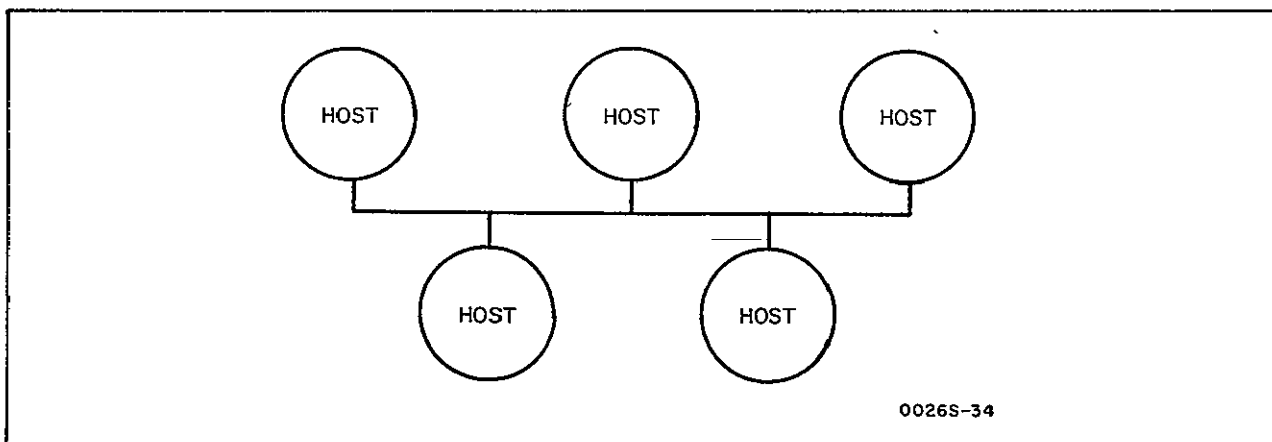


Figure 6-3. Common Bus Design

Messages are transmitted from a source host and recognized by the destination host. Access to the common bus is shared among the hosts by some allocation scheme. These schemes include simple polling, priority request, contention, and cyclic time division. In a priority request scheme, users ready to transmit make a request and are given access to the bus according to some priority method. In a contention scheme, users transmit short bursts at random and retransmit in the event that interference among two or more simultaneous transmissions occurs. Cyclic time-division schemes

dedicate particular, regularly recurring time slots to specific users for their transmissions. Hosts can readily be added to a common bus with little effect on the other hosts with a proper allocation scheme.

A potential bandwidth bottleneck exists with the common bus, and it is often impossible to increase the performance without changing the implementation of the entire bus. Failures of the bus are catastrophic.

6.2.4 CABLE BUS

The cable bus² has the same basic configuration and characteristics as the common bus. However, the cable bus uses cable television (CATV) technology and multiple-access bus techniques. In this approach, all hosts can receive all the transmitted data and, by selective filtering (using address codes or other identifiers included in each message), each host can extract from the data stream just those messages meant for him. Some cable bus schemes use frequency division multiplexing (FDM) to provide a number of logical cable buses on the same physical cable bus.

Low error rates, high throughput, and a reduced amount of wiring and communications hardware required have all been demonstrated by several cable bus systems.

To avoid catastrophic failures of the system, techniques such as dual routing of cable and redundant circuits can be employed.

Disadvantages of the cable bus are that the total capacity is limited by the bandwidth of a single bus and the cable bus is not a replacement for networks operating over distances in excess of a few miles.

6.2.5 BASIC LOOP DESIGN

Figure 6-4 shows the loop architecture, in which each host is connected to two neighboring hosts.

Messages circulate around the loop from source host to destination host with the intermediate hosts acting as store and forward units. Hosts can be inserted in the loop without significantly affecting the flow of messages.

²V. DeMarines and L. Hill, "The Cable Bus in Data Communications", Datamation, August 1976.

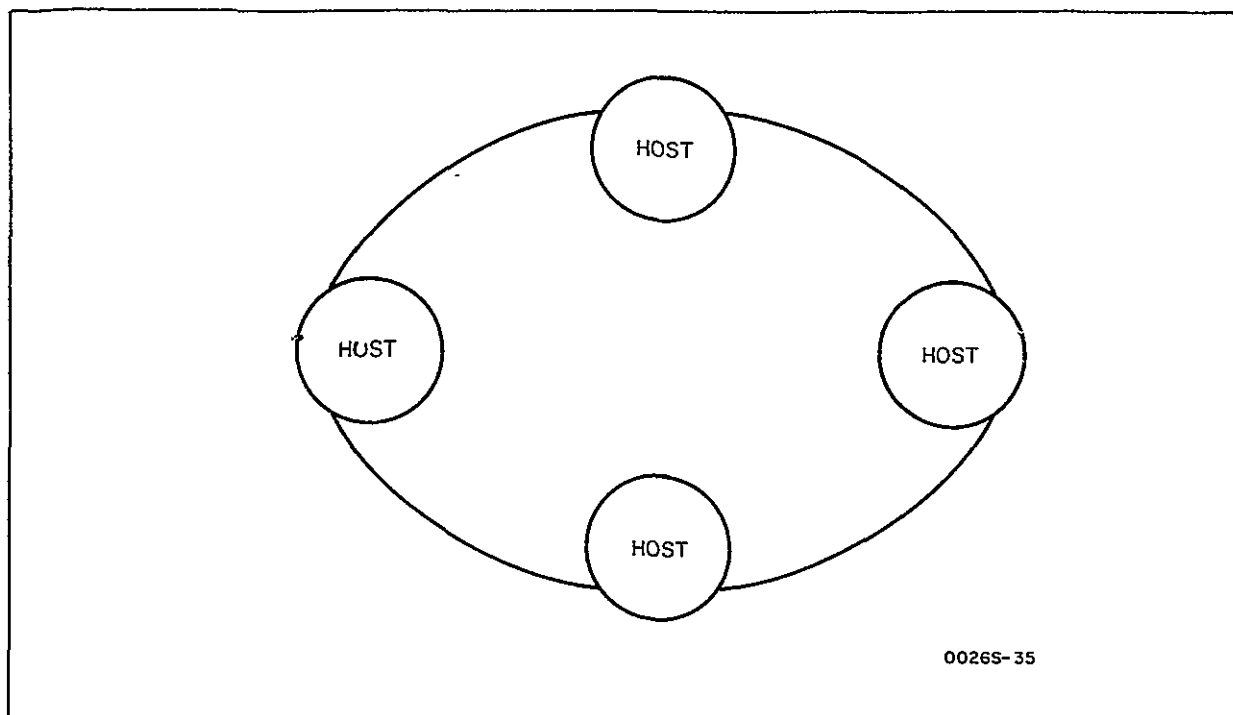


Figure 6-4. Loop Design

The failure-effect and failure-reconfiguration characteristics of the basic loop approach are poor. A single failure in either a host or an interconnecting channel can effectively impede traffic flow.

6.2.6 COMPUTER SUBNET

In this approach, a number of hosts are connected, each by a functionally single, bidirectional path to a computer subnet (see Figure 6-5). The computer subnet consists of a series of node computers.

Messages are exchanged among the hosts using the computer subnet as an intermediary. Hosts can readily be added with little effect on the other hosts. Failure-effect and failure-reconfiguration characteristics are good within the subnet, since messages can be rerouted around failures. Bandwidth problems can be minimized by implementing high-speed subnet channels and nodes with a good input/output architecture. Incremental expansion is easily accomplished at reasonable cost and without degradation of system requirements.

Faulty or malicious software in a host can be controlled if the source node monitors the expected bandwidth from the source process.

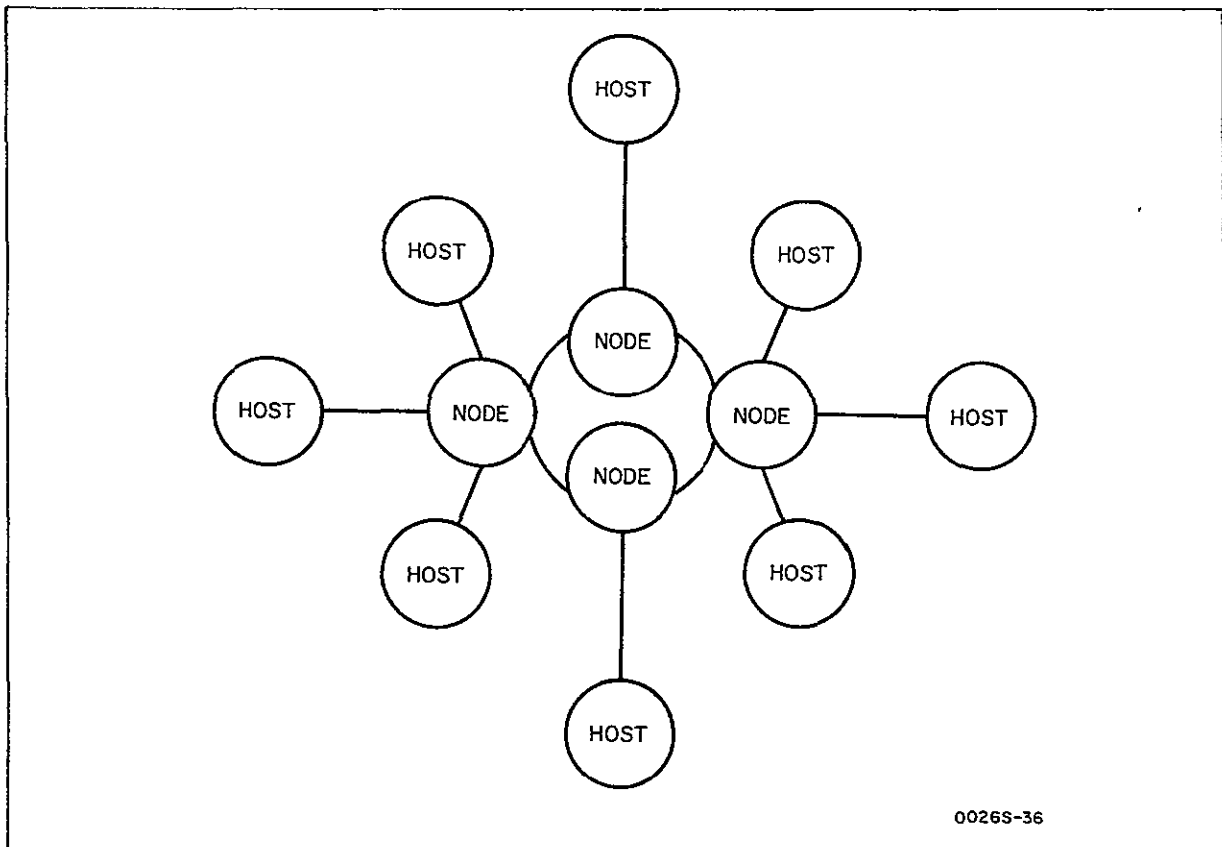


Figure 6-5. Computer Subnet

Central control of the network can be provided by utilizing one of the hosts to direct the operation of the IPC system.

The computer subnet provides more flexibility and reliability for Pocnet than the other design approaches; however, the costs of computer subnet hardware and software are also higher.

6.2.7 CONCLUSIONS

Table 6-1 illustrates the major requirements for the IPC system in relation to the design alternatives. The computer subnet is the only design which satisfies all the requirements. While it is recognized that the design modifications to some of the alternatives can alleviate certain problems, it is still felt that the computer subnet design is best suited for the IPC system. In summary, the following points are presented in favor of the computer subnet:

- a. The computer subnet approach offers existing and proven state-of-the-art technology, since similar networks have been implemented.

Table 6-1
Requirements and Design Alternatives

Requirements	Common Access Memory	Crossbar Switching	Common Bus	Cable Bus	Basic Loop	Computer Subnet
Easy to manage — Generates loading and status reports and does traces on request. Reports problems to a higher authority	No	Yes (with a DOCC host)	No	No	No	Yes (with a DOCC host)
Easy to use — Minimum or no constraint on message length or content	Yes	No	Yes	Yes	Yes	Yes
Fail soft — No failure in the network will be catastrophic	No	No	No — Dual routing or new technique required	No — Dual routing or new technique required	No	Yes
Provide real-time host communication	Yes	Yes	Yes	Yes	Yes	Yes
Expandable — Will not choke with information volume or message overhead as system is expanded. Expansion will not deteriorate system	No — Bus overload problems with a large number of hosts	No — Switch points increase geometrically with additional lines	No — Bandwidth problems with a large number of hosts	No — Bandwidth problems with a large number of hosts	No — Bandwidth problems with a large number of hosts	Yes (with proper interconnecting channels)
Manufacturer independent — Allows hosts from various manufacturers to communicate and even allows the computers making up the network to come from different sources	No — Word lengths from sources are different	Yes (with interface)	Yes (with interface)	Yes (with interface)	Yes (with interface)	Yes (with interface)
Geographically independent — Allows remote or local hosts to communicate from widely separated geographical locations. (Hundreds of feet to thousands of miles which implies serial communication)	No	No — Synchronization would be a problem	No — Common bus is usually parallel	No — Designed for local environments	No — Loop response time would be degraded	Yes

- b. Standard communication protocols exist for IPC. The use of proven standards is highly desirable since they can solidify the design and speed implementation and also prolong the life cycle.
- c. The failure-effect and failure-reconfiguration characteristics of the computer subnet are very good.

6.3 OPERATIONAL DESCRIPTION

6.3.1 OVERVIEW

The IPC system performs the basic functions required for two or more processes to exchange programs and control information. The processes are usually located in host processors connected to the Poccnet system.

Messages from one part of Poccnet find their way to other parts of Poccnet via the IPC subsystem. Figure 6-6 shows how the application processors (APs) appear with an IPC subsystem in place.

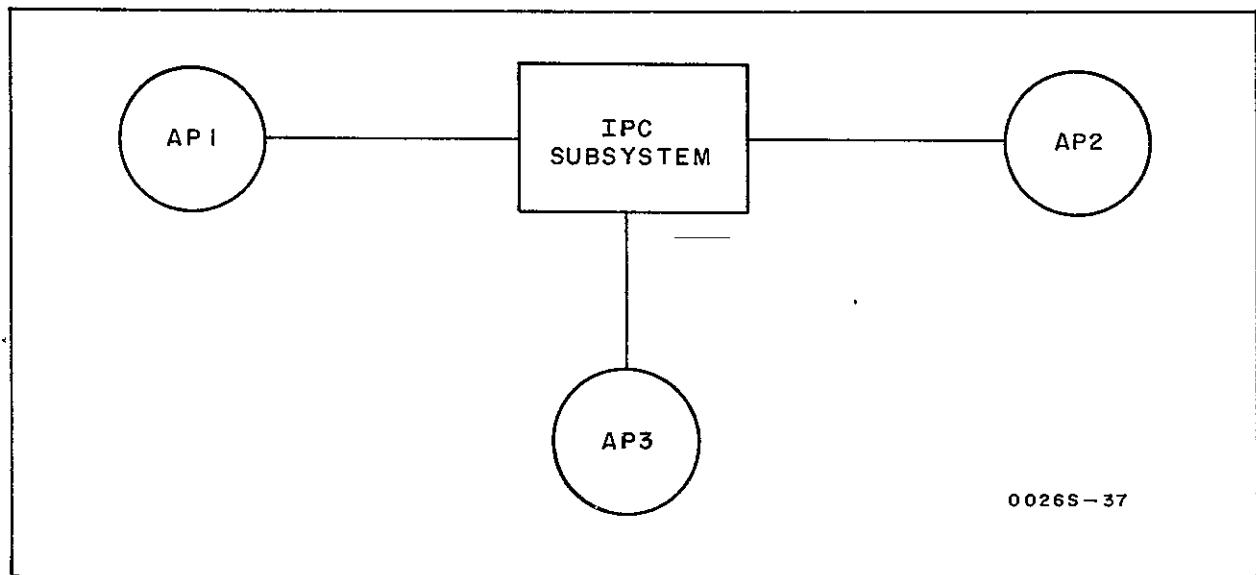


Figure 6-6. AP and IPC Subsystem

A program (or process) located in AP1 can send and receive data from a process in AP2 or AP3. Figure 6-7 shows the normal sequence of events when one process communicates with another process. Process A sends a frame and waits (or performs other tasks) for an acknowledgement. Process B receives the frame and sends the acknowledgement. Process A may now send another frame.

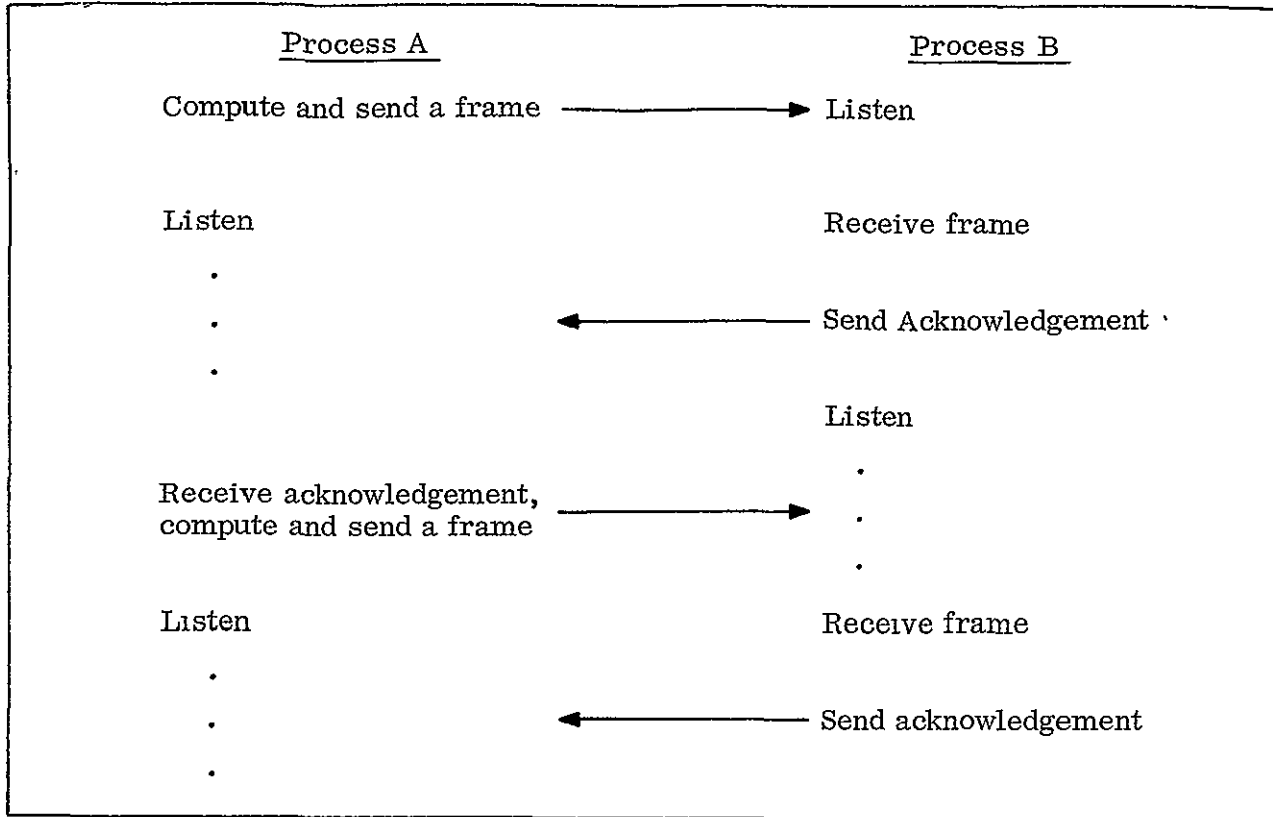


Figure 6-7. Process-to-Process Communication Sequence

Some protocols allow A to send up to some number (n) of frames before receiving an acknowledgement from B.

Two levels of frame transfers and acknowledgements exist and are shown in Figure 6-8.

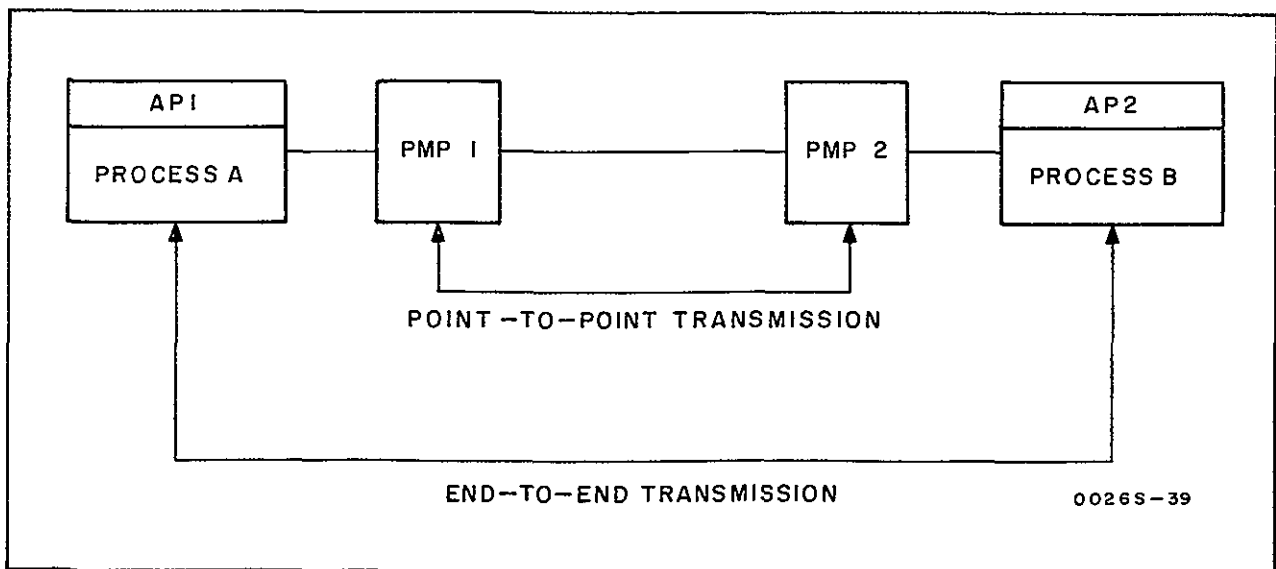


Figure 6-8. End-to-End/Point-to-Point Transmission

Point-to-point implies the transfer of frames between connected devices. End-to-end implies the transfer of frames from source to destination processes. All point-to-point transfers will be acknowledged by the receiving point (Figure 6-9).

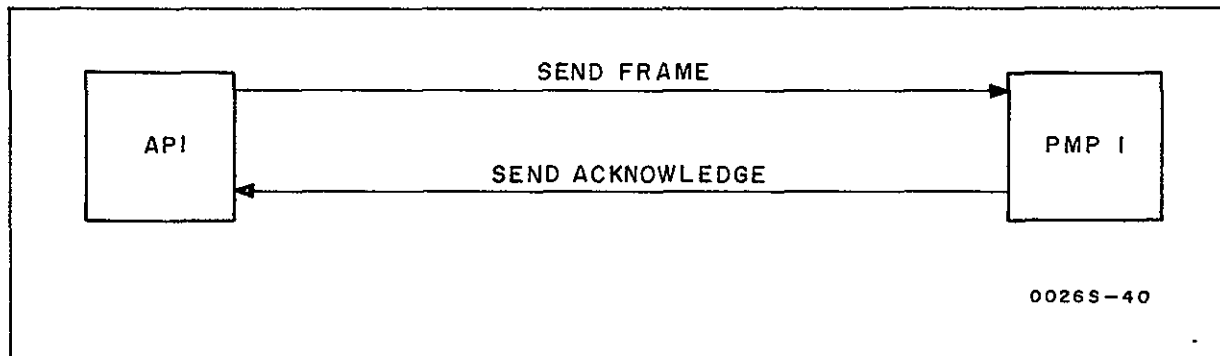


Figure 6-9. Point-to-Point Acknowledgement

The acknowledgement frequency for end-to-end transmission will be a variable. Normally, every end-to-end transfer will be acknowledged. However, when both receiving and transmitting processes agree to do so, frames can be transmitted from the sending process to the receiving process without acknowledgements. Certain real-time processes may require this capability.

6.3.2 GENERAL DESCRIPTION OF PMPNET ACCESS PROTOCOL

The conventions governing the manner in which hosts access the Poccnet message processor (PMP) subnet using a device-independent interface are specified in the Poccnet Protocols Manual which is included as Appendix 5A of this report.

6.4 POCCNET MESSAGE PROCESSOR DESCRIPTION

The PMPs are digital processors which will be used by the IPC to transfer frames through the network, keep performance statistics, and detect failures of equipment which are under PMP control. The PMPs are under the supervision of the DOCC (paragraph 6.5). Logical channels will not be constructed between processes without the authorization of the DOCC.

6.4.1 PMP CHARACTERISTICS

For this study report, the PMP is assumed to be a digital processor with the following characteristics:

- a. A main processor memory size of at least 128 16-bit kilowords with a read/write cycle time of 1 usec or less.

- b. A maximum of 10 Poccnet channels may be connected to the PMP. Each of these channels can operate at 2 megabits full duplex.
- c. Channel hardware operating in the direct-memory-access (DMA) mode with the controlling processor.

Prototype designs for these channels are discussed in Appendix 6C, Sample Channel Implementation, and in Appendix 6D, Microprocessor Implementation of X.25.

6.4.2 PMP SOFTWARE ORGANIZATION

The same PMP software will reside in each PMP and will be organized into foreground and background programs (Figure 6-10).

The foreground program will handle the receiving and transmitting of frames over the channels according to the adopted protocols. This program will use information found in the "what-to-do" table to determine mapping, frame routing, etc. The foreground program will record events in counters associated with the "what-happened" tables.

The background program will perform the bulk of the statistics keeping. The background program will read the tables, reset the counters, and respond to commands from the DOCC specifying which statistics to keep and report.

During the development and test stage, a local CRT and keyboard will be available at each PMP. The background program will accept local commands to perform the following:

- a. Display physical and logical channel statistics.
- b. Display errors and selected messages.
- c. Simulate transfers to and from the DOCC.

6.4.3 FRAME QUEUE

Each frame entering the PMP will be stored in the frame queue (Figure 6-11). Once a frame has entered the queue, the PMP will reset the associated DMA channel to receive the next frame into the next available slot in the queue. The queue will hold up to 70 frames. If the queue becomes filled, receive-not-ready messages will be transmitted to each sending PMP. Once the queue is emptied to a certain level, receive-ready messages will be transmitted to the waiting PMPs.

FOREGROUND

TABLES THAT TELL FOREGROUND
WHAT-TO-DO

- CONFIGURATION
- MAPPING
- PRIORITIES
- ROUTING
- DIAGNOSTICS
- FRAME CAPTURE/GENERATION

FOREGROUND

- PROTOCOLS
- MANAGE CHANNELS
- MANAGE BUFFERS
- UPDATE TABLES
- COPE WITH ERRORS

TABLES THAT TELL
WHAT-HAPPENED
IN FOREGROUND

- TRAFFIC
- ERROR HANDLING
- STATUS

BACKGROUND

READ OR CHANGE
WHAT-TO-DO TABLES
AND
WHAT-HAPPENED TABLES

PMP COMMAND
ENCODER / DECODER

LOCAL
CRT/KEYBOARD
SOFTWARE

TO
DOCC

0026 S-41

Figure 6-10. PMP Software Organization

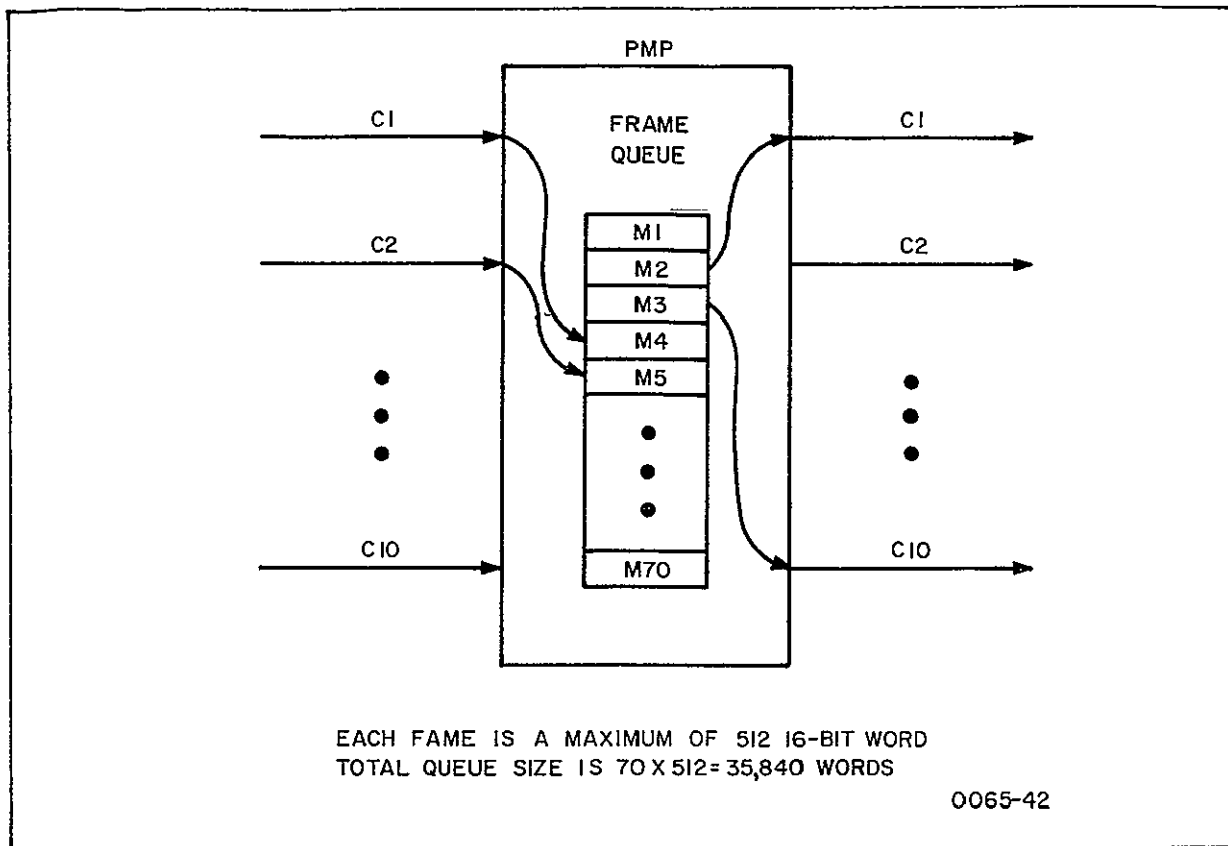


Figure 6-11. Frame Queue in the PMP

As the frames enter the queue, the PMP will determine which channel the frame is to be transmitted over and then start the DMA operation on the appropriate channel. There will be no core-to-core frame movement.

6.4.4 MESSAGE ROUTING TABLE

Each frame the PMP receives is identified by a logical channel identifier (LCI). Three separate LCIs are used. (See Figure 6-12). The source LCI is supplied by the source host and is converted by the insertion PMP to the network LCI.

The network LCI is used within the network to route the frame to the destination PMP.

The destination LCI is inserted by the destination PMP before the frame is transferred to the destination host.

Thus, the source LCI is a local identifier for the source host's process, the destination LCI is a local identifier for the destination host's process, and the network LCI is used within the network to route the frame.

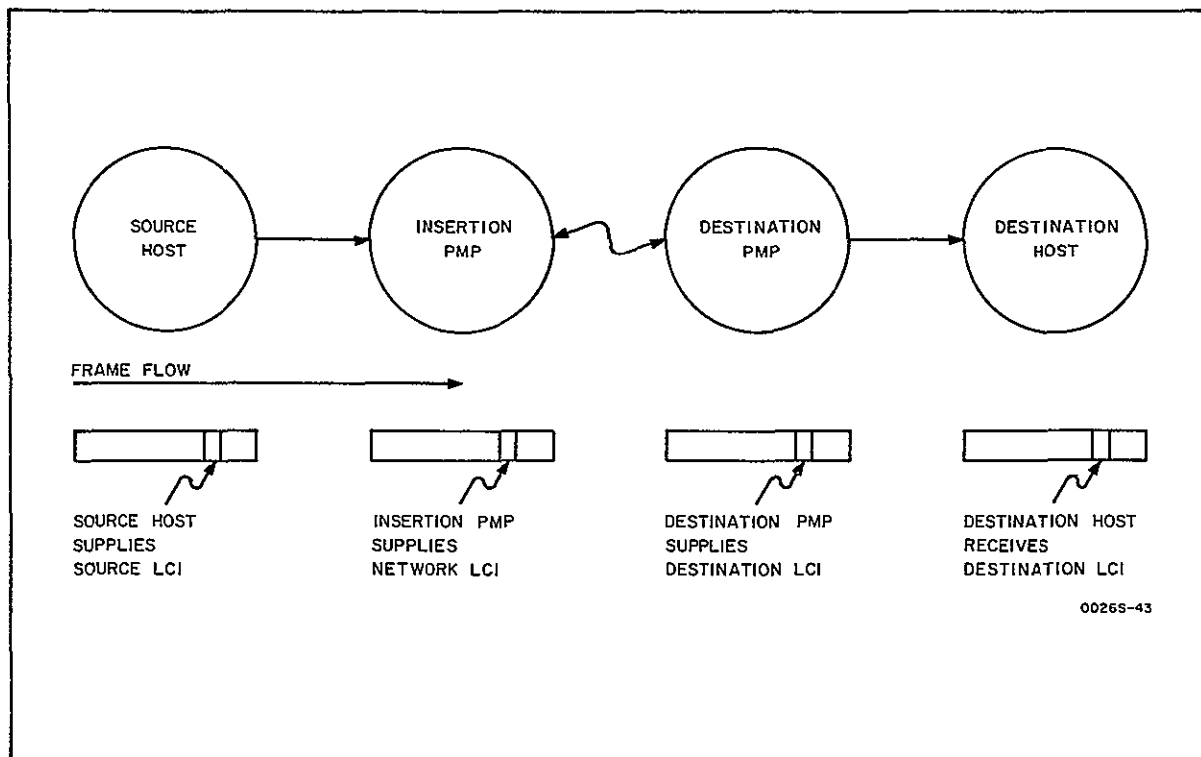


Figure 6-12. Source, Network, and Destination LCIs

Certain LCIs (source, network, and destination) are reserved for communication paths between the PMPs and DOCC. If there are n PMPs in the cluster, LCIs 0 through n handle PMP/DOCC communications. Requests to the DOCC are always made via LCI 0. Requests to PMPs 1 through n are made via LCI 1 through n .

An example of the routing tables is shown in Figure 6-13.

Two hosts (AP1 and AP2) have an established communication path via PMP1 and PMP2. The solid lines from AP2 and AP1 represent the table path in one direction and the dashed lines represent the table path in the return direction. In the example, the following steps are performed:

- A program on AP2 with a source LCI of 500 transfers a frame to PMP1.
- PMP1 receives the frame on physical channel H2 (host channel 2) and indexes into the source routing table to obtain the network LCI of 4000.
- PMP1 now examines the 4000 entry in the network/destination routing table. PMP1 determines which of the two channels the frame was received on and then transmits the frame on the other channel.

Since the frame was received on H2, the frame is transmitted on C1, C2 (the channel connecting PMP1 and PMP2).

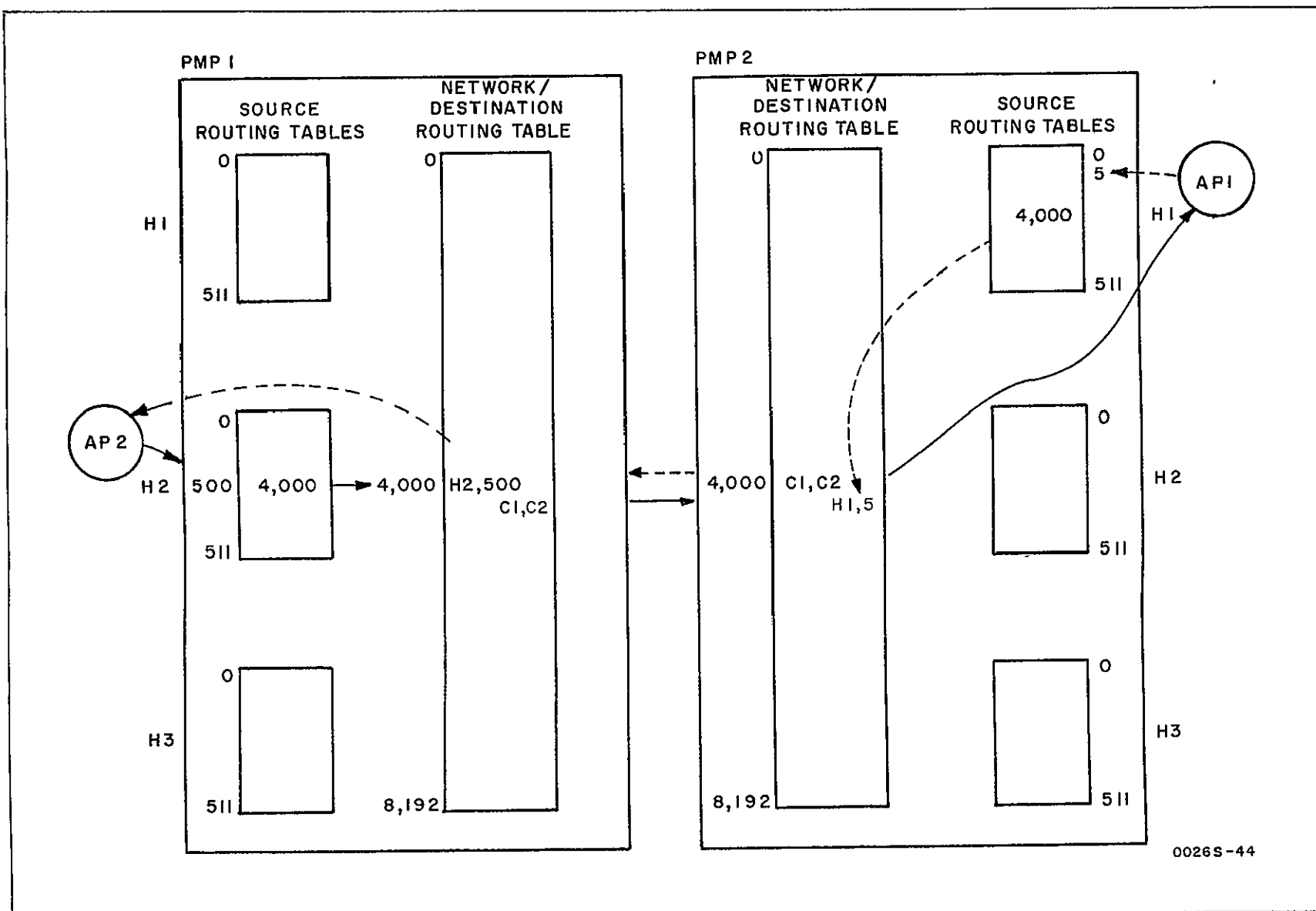


Figure 6-13. Example of Message Routing Tables

- d. PMP2 receives the frame on C1, C2, and transmits it on H1. PMP2 also inserts the destination LCI of 5 into the frame.
- e. AP1 receives the frame for the program with a destination LCI of 5.
- f. AP1 transmits a frame with a source LCI of 5 to PMP2.
- g. PMP2 receives the frame on physical channel H1 (host channel 1), indexes into the source routing table to obtain the network LCI of 4000, and transmits the frame over C1, C2.
- h. PMP1 receives the frame on C1, C2, inserts a source LCI of 500, and transmits the frame to H2.
- i. AP2 receives the frame for the program with a destination LCI of 500.

6.4.5 PMP STATISTICS

Each PMP will maintain a traffic by physical channel table and a traffic by LCI table. These tables permit the PMP to monitor both physical and logical channels.

6.4.5.1 Traffic by Physical Channel. Figure 6-14 shows an example of the traffic-by-physical-channel table in a PMP (such as PMP4).

PHYSICAL CHANNEL	TOTAL FRAMES	INPUT FRAME COUNT	AVERAGE INPUT LENGTH	POLY ERRORS	OUTPUT FRAME COUNT	AVERAGE OUTPUT LENGTH	RE-XMIT
C4,5	1,222	1,000	8,000	16	222	40	0
C4,6	6	1	40	0	5	40	0
C4,7	2,355	5	40	0	2350	8192	10
C4,8	17	8	40	0	9	40	0

Figure 6-14. Sample Traffic-by-Physical-Channel Table

The headers for the above figure have the following meanings:

- a. Physical channel — The physical channel connected to the PMP.
- b. Total frames — The total frames (both received and transmitted) logged on the channel.
- c. Input frame count — The total frames received on this channel.
- d. Average input length — The average input length of the total frames received.

- c. Poly errors — The number of received frames that had a polynomial error.
- f. Output frame count — The total frames transmitted on this channel.
- g. Average output length — The average output length of the total frames transmitted.
- h. RE-XMIT — The total number of frames which had to be retransmitted.

6.4.5.2 Traffic by LCI. Each process-to-process path is identified by a unique network LCI number. Associated with each unique network LCI are two physical channels on each PMP through which the logical channel passes. The frame is received on one physical channel and transmitted on the other physical channel for one flow direction and is reversed for the other flow direction. An example of the traffic-by-LCI table is shown in Figure 6-15.

The headers for Figure 6-15 have the following meanings:

- a. LCI — The network logical channel identifier.
- b. PC — The first physical channels associated with this LCI. The next six columns represent the counts for this PC.
- c. Input count — The total frames received on the PC.
- d. Ave. input len — The average input length of the total frames received.
- e. Poly error — The number of received frames which had a polynomial error.
- f. Out count — The total frames transmitted on this PC.
- g. Ave. out len — The average output length of the total frames transmitted on this PC.
- h. RXMIT — The total number of frames which had to be retransmitted.
- i. PC — The second physical channel associated with this LCI. The next six columns represent the counts for this PC. The column headers have the same meaning as previously described.

The PMP will transfer the frame statistics to the DOCC and then reset the counts when the DOCC requests the statistics or when the counts reach a certain threshold.

<u>LCI</u>	<u>PC</u>	<u>INPUT</u> <u>COUNT</u>	<u>AVE.</u> <u>INPUT</u> <u>LEN</u>	<u>POLY</u> <u>ERROR</u>	<u>OUT</u> <u>COUNT</u>	<u>AVE</u> <u>OUT</u> <u>LEN</u>	<u>RXMIT</u>	<u>PC</u>	<u>INPUT</u> <u>COUNT</u>	<u>AVE</u> <u>INPUT</u> <u>LEN</u>	<u>POLY</u> <u>ERROR</u>	<u>OUTPUT</u> <u>COUNT</u>	<u>AVE</u> <u>OUT</u> <u>LEN</u>	<u>RXMIT</u>
0														
1														
2														
.														
.														
.														
100	C4,5	900	8000	3	10	40	0	C4,6	10	40	1	900	8000	6
.														
.														
.														
4095														

Figure 6-15. Sample Traffic-by-LCI Table

6.4.6 PMP TIMING ESTIMATE

This section presents an initial estimate of the PMP timing.

At the 2-megabit rate, a 16-bit word enters the PMP memory every 8 usec. The bus overhead per 512-word message is as follows:

bus overhead = bus latency time + memory cycle time

$$\text{bus overhead} = \left[2.7 \text{ usec} * \frac{512 \text{ words}}{16 \text{ words}} \right] + \left[512 \text{ words} * \frac{.98 \text{ usec}}{\text{word}} \right] = 588.2 \text{ usec}$$

Generally accepted figures are used in this equation. The bus latency figure of 2.7 usec occurs every 16 words (a 16-word FIFO buffer is assumed). A memory cycle speed of 0.980 usec is also assumed. The percentage of time that the CPU is dedicated to the bus is the bus overhead divided by the total transfer time.

$$\text{Percent of time the CPU is dedicated to the bus} = \frac{588.2}{512 * 8} = 14.4\%$$

Table 6-2 shows the CPU degradation caused by active DMA channels. From the table, note that the PMP cannot handle seven or more channels transmitting simultaneously (worst case) because of the CPU degradation caused by I/O bus latency. The CPU becomes completely locked out when the number of DMA channels is seven or greater. In this case, the PMP doing the transmitting will not receive an acknowledgement and will retransmit the message.

Table 6-2
CPU Degradation

Number of DMAs Active	Percent of time the CPU is dedicated to the bus (cycle stealing)
0	0
1	14.4
2	28.8
3	43.2
4	57.6
5	72.0
6	86.4
7	100.8 (data lost)
8	Data lost
9	Data lost
10	Data lost

The input interrupt routine requires less than 100 usec to complete and consists of the following steps:

- a. Save registers.
- b. Mark message arrived.
- c. Rearm the DMA input operation to receive next message into the next available queue slot.
- d. If queue is full, inform level 2 to send receive-not-ready message.
- e. Reload registers and return to the interrupted program.

The main processing loop requires less than 300 usec to complete and consists of the following steps:

- a. When mark message has arrived, go to step c; otherwise, go to step b.
- b. Update the CRT, process operator inputs, update queue points relating to messages, go to step a.
- c. Use the LCI to determine the output channel and start the DMA output operation.
- d. Update statistics, queue pointers, and return to step a.

The output interrupt routine requires less than 100 usec to complete and consists of the following steps:

- a. Save registers.
- b. Mark message sent.
- c. Reload registers and return to the interrupted program.

For worst-case analysis, assume 10 messages enter the PMP simultaneously at an average rate of 1 megabit (which is acceptable).

Other throughput considerations are discussed in Appendix 6H, High Performance Local Communications, based on the CCITT X.25 protocol.

The message input interrupt routine requires 100 usec/channel, 300 usec/channel for processing, and 100 usec/channel to handle the DMA output.

10 channels	*	100 usec for input interrupt	=	1 msec
10 channels	*	300 usec for processing	=	3 msec
10 channels	*	100 usec for output interrupt	=	<u>1 msec</u>
				5 msec

That is, it will take 5 msec of processing to completely clear out the ten simultaneous input frames. This assumes that, while the PMP is performing the DMA transmit operations, none of the 10 input channels becomes active since data would be lost due to CPU overload. Furthermore, the CPU degradation due to the DMA activity must be added to the 5-ms timing figure. Therefore, the PMP software is not the limiting factor, but the I/O bus overhead is.

As described in paragraph 4.3.1, a detailed simulation of the proposed PMP subnet configuration was run with various realistic and unrealistic host traffic models. It was found that with realistic traffic the PMPs were very lightly loaded. Even with unrealistically heavy traffic, the nature of PMP degradation was such that a private CPU bus would relieve the congestion. More details are given in Appendix 4B.

6.4.7 PMP CORE ESTIMATE

The initial PMP core estimate is 88 16-bit kilowords. The program size is 20 kilowords with 68 kilowords of buffers. The various portions of core are dedicated as follows:

Operating system	=	12 kilowords
PMP program	=	8 kilowords
Frame queue	=	36 kilowords
Source routing table (16 hosts/PMP * 512)	=	8 kilowords
Network/destination routing table	=	8 kilowords
Statistics tables	=	<u>16 kilowords</u>
Total core	≈	88 kilowords

6.5 IPC DOCC DESCRIPTION

The DOCC can be viewed as the system partitioner and is responsible for permitting the allocation of computer resources and configuring them into partitions.

Logically, the DOCC software can be divided into two main tasks, the IPC DOCC and the host DOCC (Figure 6-16).

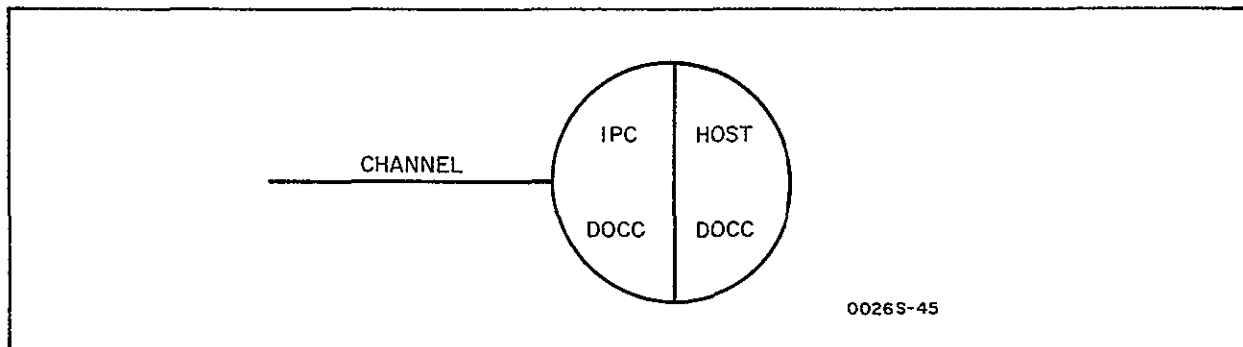


Figure 6-16. IPC/HOST DOCC

The host DOCC performs high-level tasks such as maintaining alphanumeric resource directories and handling mnemonic requests from resources which desire to establish or discontinue a communication path. Suggested user-level commands which the host DOCC would accept and process are discussed in paragraph 6.7.

The IPC DOCC is a complete layer that functions without a host DOCC. Basically, the IPC DOCC performs low-level tasks such as monitoring physical channel activity, maintaining a record of all logical circuits in the network, and handling absolute requests from resources which desire to establish or discontinue a communication path.

The goal is to provide a minimum IPC DOCC task which can be augmented with higher layers (such as the host DOCC) in the future. An example of the IPC DOCC command language is illustrated in Appendix 6E. Network control issues are discussed in Appendix 6H.

6.6 ESTABLISHMENT OF A MINIMUM POCCNET SYSTEM

In order to initialize the DOCC and the PMPs and establish a minimum Poccnet system, the following functions would be performed:

- a. Load the PMP operating system into each PMP and assign physical names to each PMP and the channels which are directly connected to each PMP.
- b. Load the DOCC operating system into the DOCC and define the cluster configuration. The DOCC must compute frame routing tables for each PMP and transfer each table to the correct PMP.

- c. Establish a minimum operating environment by having the hosts define themselves and their resources.

Details of these functions are described in Appendix 6F.

6.7 USER-LEVEL COMMANDS

This section contains suggestions generated by the Poccnet IPC Subsystem Committee and presented during interface meetings with other Poccnet subsystem committees. These suggestions relate to the manner in which host operators and/or host processes (programs) might make high-level requests on their host software system to request interprocess communication. The host software system would then convert these high-level requests into the actual packets required to interface with the IPC system.

The suggestions for host operator requests involve the extensions of the Systems Test and Operation Language (STOL) to accept manual or automated directives such as CONNECT directives to establish an actual connection between two processes and SEND directives to transmit messages via an established connection. The STOL language processor would recognize directives of this nature and activate the associated software routines to convert these operator requests into the necessary packets to interface into the IPC system.

The suggestions for host program requests involve the implementation of FORTRAN library routines that are callable by programs via CALL CONNCT, CALL SEND, CALL RECEVE directives, etc. These calls would activate the associated software routines to convert the requests into an interface with the IPC system.

In fact, the STOL IPC extensions would probably most efficiently be implemented by converting the associated directives into these same FORTRAN calls, and, consequently, the STOL extensions would execute as any other host program as far as requests for interprocess communication. Appendix 6G describes the details of the user-level commands.

SECTION 7
APPLICATIONS ENGINEERING

SECTION 7

APPLICATIONS ENGINEERING

CONTENTS

<u>Paragraph</u>		<u>Page</u>
7.1	General	7-1
7.2	POCC Applications	7-1
7.3	Data Base Storage Applications	7-2
7.3.1	Data, Catalog, and Library Management and Main- tenance	7-3
7.3.2	Software Engineering Management and Configuration Management Tools	7-3
7.3.3	Mission Planning Tools	7-3
7.3.4	Document Preparation and Maintenance Tools	7-8
7.3.5	Computer Conference (Mailbox Service)	7-9
7.4	Simulation	7-9
7.4.1	Requirements	7-10
7.4.1.1	Training	7-10
7.4.1.2	Operations	7-10
7.4.1.3	Analysis	7-11
7.4.2	Design Goals	7-11
7.4.2.1	Payload Simulation/POCC Interfaces	7-11
7.4.2.2	Simulator Installation	7-13
7.4.2.3	On-Line Operations	7-13
7.4.2.4	Off-Line Operations	7-13
7.4.2.5	Resource Sharing and Modularity	7-14
7.4.3	Functional Design of Payload Simulators	7-14
7.4.4	Operational Capabilities	7-18
7.4.4.1	Real-Time and Non-Real-Time Operation	7-18
7.4.4.2	On-Line and Off-Line Configurations	7-19
7.4.4.3	Phases and Modes	7-19
7.4.4.4	Checkpointing and Refreshing	7-20
7.4.4.5	Interactive Control via Simulation Director Console	7-21

CONTENTS (Cont)

<u>Paragraph</u>		<u>Page</u>
7.4.4.6	System Timing and Diagnostics	7-21
7.5	On-Board Computer Support	7-22
7.5.1	On-Line OBC Support	7-22
7.5.2	Off-Line OBC Support	7-23
7.6	Command Memory Management (CMM)	7-24
7.6.1	Interface Information	7-24
7.6.2	Memory Load Structure Information	7-25
7.6.3	Command Loading Information	7-25
7.6.4	Clock Information	7-25
7.6.5	Operational Modes Information	7-26

ILLUSTRATIONS

<u>Figure</u>		<u>Page</u>
7-1	POCC Mission Planning Interfaces	7-5
7-2	Payload Simulation/POCC Interfaces	7-12
7-3	Payload Simulations System Structure	7-15

TABLES

<u>Table</u>		<u>Page</u>
7-1	Summary of Simulation Types Versus Simulation Components and Characteristics	7-17

SECTION 7

APPLICATIONS ENGINEERING

7.1 GENERAL

The purpose of Pocnet is to provide standard POCCs and standard POCC spacecraft control computing support and interfaces by performing the common systems engineering once for all GSFC POCCs, then providing the required Pocnet systems for building Model POCCs with common software modules. Individual POCC spacecraft control computing and data base storage services are provided under an activity called applications engineering. This section includes a discussion of the following five major applications engineering areas: POCC applications, DBS applications, simulation, OBC support, and command memory management.

7.2 POCC APPLICATIONS

The following paragraphs outline an approach for developing POCC applications for each mission. This development process makes use of the major resources of the Pocnet software engineering methodology, the Model POCC, and the repository for common POCC software.

The Pocnet software engineering methodology provides a flexible standard approach to the overall software system engineering problem which can be tailored to individual project needs and requirements. This standard approach includes specifications for a phased development cycle, top-down design and phase implementation, use of software development tools, personnel training guidelines, quality assurance guidelines, configuration management, and validation and verification procedures.

The Model POCC activity serves three basic purposes. First, it serves to identify and describe POCC functional capabilities and requirements. Second, the Model POCC acts as a driver for design by identifying candidate software for modularization and standardization. Third, the Model POCC serves as a tool for familiarization and planning.

The repository for common POCC software is a collection of relevant programs and data pertaining to software identified as being good candidates for modularization and/or standardization in POCC development. It is intended to be a prime source of design and implementation modules and ideas for software packages typically found in POCCs.

In the Pocnet era, it is anticipated that a POCC designer and/or implementer will proceed with his activities in basically the following general stages:

- a. Adopt a version of the Pocnet software engineering methodology tailored to his specific needs.
- b. Formulate a complete and comprehensive statement of his problem. This includes generating an exhaustive and unambiguous statement of project requirements.
- c. Make a comparison between what he needs and what the Model POCC offers and note the variances. (These variances should correspond to his mission uniques.)
- d. Utilize to the maximum extent possible the design and/or implementation modules and information contained in the repository to achieve his objectives.
- e. Engineer the truly mission-unique software following the tailored methodology adopted.

If the three major resources are utilized to the fullest extent, it is to be expected that applications engineered in this manner will have very high reliability and will be achieved with substantial savings in manpower, time, and money.

7.3 DATA BASE STORAGE APPLICATIONS

The DBS system is primarily a storage and retrieval facility for data, but it will also offer a small number of basic services related to its primary objective. These services are provided in the form of applications programs which run on the DBS computers or on other applications processors on or off Pocnet. These applications include the following:

- a. Data, catalog, and library management and maintenance.
- b. Software engineering management and configuration management tools.
- c. Document preparation and maintenance.
- d. Computer conferencing (mailbox services).

7.3.1 DATA, CATALOG, AND LIBRARY MANAGEMENT AND MAINTENANCE

Applications programs will be provided to aid users in the management of data bases and program libraries. Programs will be provided to build, modify and delete catalogs and to produce listings of the names and characteristics of the programs or data files. Additional programs will be available to list the contents of data files or to list programs.

An important and frequently occurring task performed on Poccnet will be the transfer of an entire data or program file from the DBS to another host in Poccnet. To accomplish this task efficiently but without putting an unreasonable burden on the user, a pair of applications programs (one in the DBS and one in the receiving host) would be used. The DBS process would block records and send out these blocks as the receiving process requests them. The receiving process would call for data blocks from the DBS, deblock records, and supply them to the user. This situation could, of course, be reversed so that a user could transfer a file from a host computer through IPC into the DBS. These programs could also be used to transfer files or entire data bases to or from off-line backup storage. The commands used to transfer data between processes are specified by the Poccnet File Transfer Protocol (PFTP), which is described in paragraph 7.1 of the Poccnet Protocols Manual (Appendix 5A).

7.3.2 SOFTWARE ENGINEERING MANAGEMENT AND CONFIGURATION MANAGEMENT TOOLS

DBS system support of software engineering management and configuration management will be provided mainly by the facilities for the creation and maintenance of data bases and by report generation software. Using these facilities, relevant information can be maintained on line and can be used to generate reports showing timelines, adherence to schedule, unresolved trouble reports, and other information required by management.

7.3.3 MISSION PLANNING TOOLS

An important tool that will be supported by the DBS system is the POCC mission planning system (MPS). The data bases used by this system will reside on the DBS system. Some of the applications processes required to run this system will execute on the DBS system while others will run in the POCC APs or in support computers.

The POCC MPS is an interactive computer system that will be developed for each POCC to assist the POCC and project mission planners in planning the operation of the spacecraft and the POCC. Mission planning is the merging of the experiment and spacecraft planning activities into operational requirements that result in a unified set of operations plans, instructions, and spacecraft operations which are intended to result in the acquisition of the required data or spacecraft performance to achieve mission and experiment objectives. A significant amount of the mission planning in the 1980s will take place within the POCCs.

As shown in Figure 7-1, a POCC MPS has a number of interfaces which must be factored into the requirements and design. Planning variables of interest to a POCC mission planner are many and diverse, and their interrelationships are frequently fluid. Some of these variables are as follows:

- a. Spacecraft Items (Orbital/Geometric Constraints)
 - (1) Attitude.
 - (2) Range.
 - (3) Location (South Atlantic Anomaly, etc.).
 - (4) Altitude (apogee, perigee).
 - (5) Sunlight (earth or lunar shadow).
 - (6) Relative sun angle.
 - (7) Relative location to another spacecraft (mutual data coverage).
 - (8) Orbit stability.
 - (9) Maneuver requirements.
- b. Spacecraft Visibility Items
 - (1) STDN visibility (TDRS or GSTDN).
 - (2) AOS/LOS times.
 - (3) Duration.
 - (4) Antenna masks/patterns.
 - (5) Elevation.

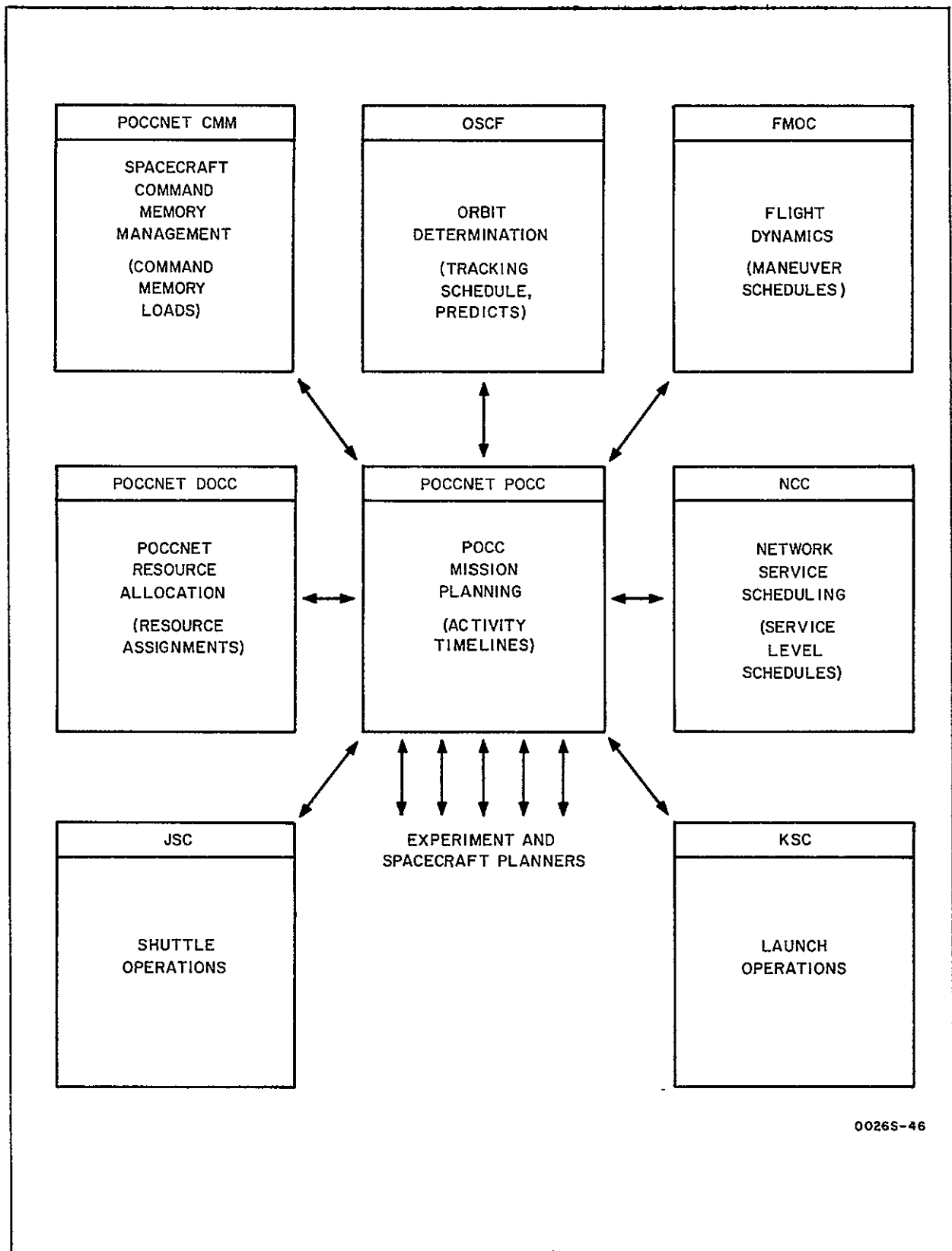


Figure 7-1. POCC Mission Planning Interfaces

- (6) Time since last support.
 - (7) Handover view periods.
 - (8) Optimum coverage zones.
- c. Spacecraft Hardware Constraints
- (1) Tape recorder capacity (or other data storage device) — Time per recorder, number of recorders.
 - (2) Power budget, thermal budget.
 - (3) OBC capability and capacity — Spacing of loads (e.g., command memory, orbit data) depends on how long a load will last which depends on spacecraft activities.
 - (4) Spacecraft health parameters and service requirements.
 - (5) Payload status, experiment status.
- d. Spacecraft Operations
- (1) Experimenter requirements (multiple).
 - (a) Verbal, TTY, interactive CRT, cards, listings, etc.
 - (b) Normal (internal conflicts and priorities).
 - (c) Unusual (solar flares, hurricane, earthquake, comet, volcano, etc.).
 - (2) Results of a previous observation (turnaround of science evaluation).
 - (3) Payload control (flight maneuver requirements).
 - (4) Payload and experiment performance (verification of OBC loads, experiment settings, etc.).
 - (5) People availability.
 - (6) POCC capability/availability.
 - (7) Support computing capability/availability (attitude, orbit, CMM).
 - (8) Shuttle timeline.

One of the outputs of mission planning is a list of required resources for total ground system support of the operations planned. This includes the utilization of POCC, Poccnet, OSCF, STDN, NASCOM, and other GSFC or NASA resources.

The MPS will be required to perform the following functions:

- a. Access the experiment planning data base to obtain or store experiment planning information (command sequences, special instructions, constraints, etc.).
- b. Access the spacecraft planning data base for payload operations ground rules, constraints, procedures, etc.
- c. Access other data bases for information (such as SMCC shuttle timeline).
- d. Access the Poccnet operational data base for use in resource availability and for requesting support.
- e. Access the command memory management system to build and store mission operation sequences and command requests and to load command memory.
- f. Access the orbit support computing facility to obtain support or information such as ephemeris, elements, or scheduling aids.
- g. Access the Flight Maneuver Operations Center to coordinate support, obtain maneuver schedules, or command requests.
- h. Access and interface with the Network Control Center. It is envisioned that the MPS will provide the interface with the Network Control Center Scheduling System.

The POCC will be tied in with the NASCOM/STDN scheduling system in real-time via a CRT/keyboard terminal and high-speed printer. Information to be transmitted to the POCC will consist of NASCOM/STDN facility status; confirmed POCC schedules for lines, stations, and service levels; payload priority information; and network forecasts. POCC information to be transmitted to the network will include POCC schedule requests, POCC status, resolutions of schedule conflicts, requests for critical support, requests for support during a spacecraft emergency, acknowledgments, ground configuration directives, and ground configuration directive requests. During periods that the POCC is not manned or the interactive terminal is allocated to a function other than STDN scheduling, the DBS system will act as the interface (mailbox)

for handshaking and message processing. The STDN scheduling system will interface to Poccnet through a gateway using the NASCOM 4800-bit block format.

The MPS will be interactive to the POCC and to the DOCC. The DOCC will maintain a set of common data base files for reference use by the POCCs. The files will contain information such as the spacecraft ephemeris (elements, vectors, or predicts), the latest SMCC shuttle timeline, or the schedule for commonly used resources. The MPS is to be primarily a tool to access and manage information and programs for the mission planners use and a tool to institute the transfer of the resulting information so that it can be implemented (schedule input requests to the NCC, etc.).

A set of normalized support level generics will be maintained in the MPS for use by the POCC or DOCC to run for mission-support analysis, conflict studies, etc. These generics can also be used for compiling the nominal support requirements for a spacecraft and for reviewing its adequacy as part of the input preparation to the NCC scheduling system. The input to the NCC scheduling system for the initial planning period will be either the approved generic or the resultant set of specific support requests. Additional generics will be tailored for specific spacecraft use and stored for use in planning operations.

Mission planning programs that can be tailored to be of use to more than one POCC or spacecraft will be developed or obtained for use via the common software modules program. An example of a program that falls in this category is a tape recorder management program that can support numerous POCCs.

7.3.4 DOCUMENT PREPARATION AND MAINTENANCE TOOLS

DBS system applications will be provided to support the generation and maintenance of documents related to POCC development and operation and will include the following:

- a. A general text editor capable of processing text consisting of both uppercase and lowercase characters.
- b. Output formatting programs which can number pages, adjust margins, and interpret control characters to provide underlining, indentation, special fonts, overstriking, or spacing for figures.
- c. Programs which can automatically generate a table of contents, list of illustrations, and list of acronyms.

7.3.5 COMPUTER CONFERENCING (MAILBOX SERVICE)

The DBS systems will host applications programs which enable users to exchange messages via the communications subnet. One program will accept incoming messages and queue them for the intended users. Another program will accept and process requests to read previously queued messages. Message queues will be maintained not only for individual users but also for groups of users with common interests (such as all HEAO operations personnel).

7.4 SIMULATION

The following paragraphs describe a general approach to payload and POCC subsystem simulations applicable to the Poccnet environment at GSFC.

In the past, spacecraft simulators have proven to be powerful tools in many respects. Operations control center personnel training, control center software checkout, mission simulation, and analysis are the principal areas where such simulators have been particularly valuable. ATSSIM, the real-time dynamic simulator for ATS-6 developed at GSFC during 1973-74, is a recent case in point. Currently SIMIUE, a real-time dynamic simulator for IUE, is in development. ATSSIM and SIMIUE were developed primarily for the purpose of training control center personnel, with ground software checkout and human engineering evaluation as secondary goals.

The following types of payload simulations* are of interest to the various users at GSFC:

- a. Training/flight plan.
- b. Analysis (mainly attitude control subsystem).
- c. On-board computer (OBC) rewrite validation.
- d. Telemetry displays validation.
- e. Command validation.
- f. Experiment.

*A detailed discussion on payload simulation types can be found in Chapter 5 of "Poccnet Studies Interim Report" prepared for NASA GSFC by the OAO Corporation, November 1976. This chapter appears as Appendix 7A of this report.

- g. Attitude determination software validation.
- h. Command memory management software validation.
- i. Power management software validation.

As described in paragraph 7.4.3, these payload simulation types constitute a family in which a common need of several members can often be satisfied by one functional module.

In the Poccnet environment, there is the additional need for POCC subsystem simulators (for example, telemetry and command system simulators and gateway simulators). The Poccnet systems engineering requirement is that each of these systems must be able to simulate itself in the sense of generating simulated traffic.

7.4.1 REQUIREMENTS

There are three categories of mission support drivers for simulations: training, operations, and analysis. Paragraph 7.4.1.1 through 7.4.1.3 describe these in detail.

Just as the requirements for modularity, reusability, and resource sharing are of paramount importance in developing the spacecraft and POCC for the 1980s, so they are also generic requirements for development of all support software. Therefore, modularity, reusability, and resource sharing are to be considered as part of the overall requirements for the payload and POCC subsystem simulations.

7.4.1.1 Training. The training requirement is to provide interactive, real-time, dynamic simulations of spacecraft systems and subsystems to serve as realistic vehicles for the following:

- a. Training POCC personnel in commanding the spacecraft.
- b. Training ground station personnel in commanding the spacecraft.
- c. Training experimenters in the mechanics of conducting on-board experiments.

7.4.1.2 Operations. The operations requirement is to provide simulations to aid in the checkout of operational software/hardware at the POCC. Specifically, simulations should assist in the following:

- a. Testing operational software.

- b. Developing and testing operational procedures.
- c. Verifying adequacy of software/hardware configuration.
- d. Evaluating efficiency and efficacy of displays.
- e. Validating block data uplinks and procedures.
- f. Validating special POCC software for attitude determination, command memory management, and power management.
- g. Validating spacecraft recovery modules and procedures.

7.4.1.3 Analysis. The analysis requirement is to provide simulations for the following:

- a. Analysis of the attitude control system and spacecraft subsystems to serve as a design evaluation tool during the initial phases of a mission.
- b. Postlaunch analysis capability to handle changes in spacecraft configuration which are a result of degradation or failures.
- c. Validating proposed OBC modifications.

7.4.2 DESIGN GOALS

It is anticipated that payload simulations will run on Pocnet host computers or on computers connected to the POCC via NASCOM. Furthermore, the host computers will sometimes not be dedicated to the simulations but will run in a multiprogramming environment. This can place a severe constraint on a simulator which is required to run in real-time. In addition, the requirements for maximum telemetry and commanding rates, integration step size for the dynamic equations, etc., must be carefully evaluated in each case to determine simulation feasibility.

The discussion that follows is generally applicable to all members in the family of payload simulators, although details will vary from one type of simulator to another.

7.4.2.1 Payload Simulation/POCC Interfaces. The payload simulation/POCC interfaces are shown in Figure 7-2. Note that two different alternative types of interface are shown. The upper path, via NASCOM interface, connects the simulation computer to the POCC front-end system, the TAC. This corresponds to the "all-up" simulation

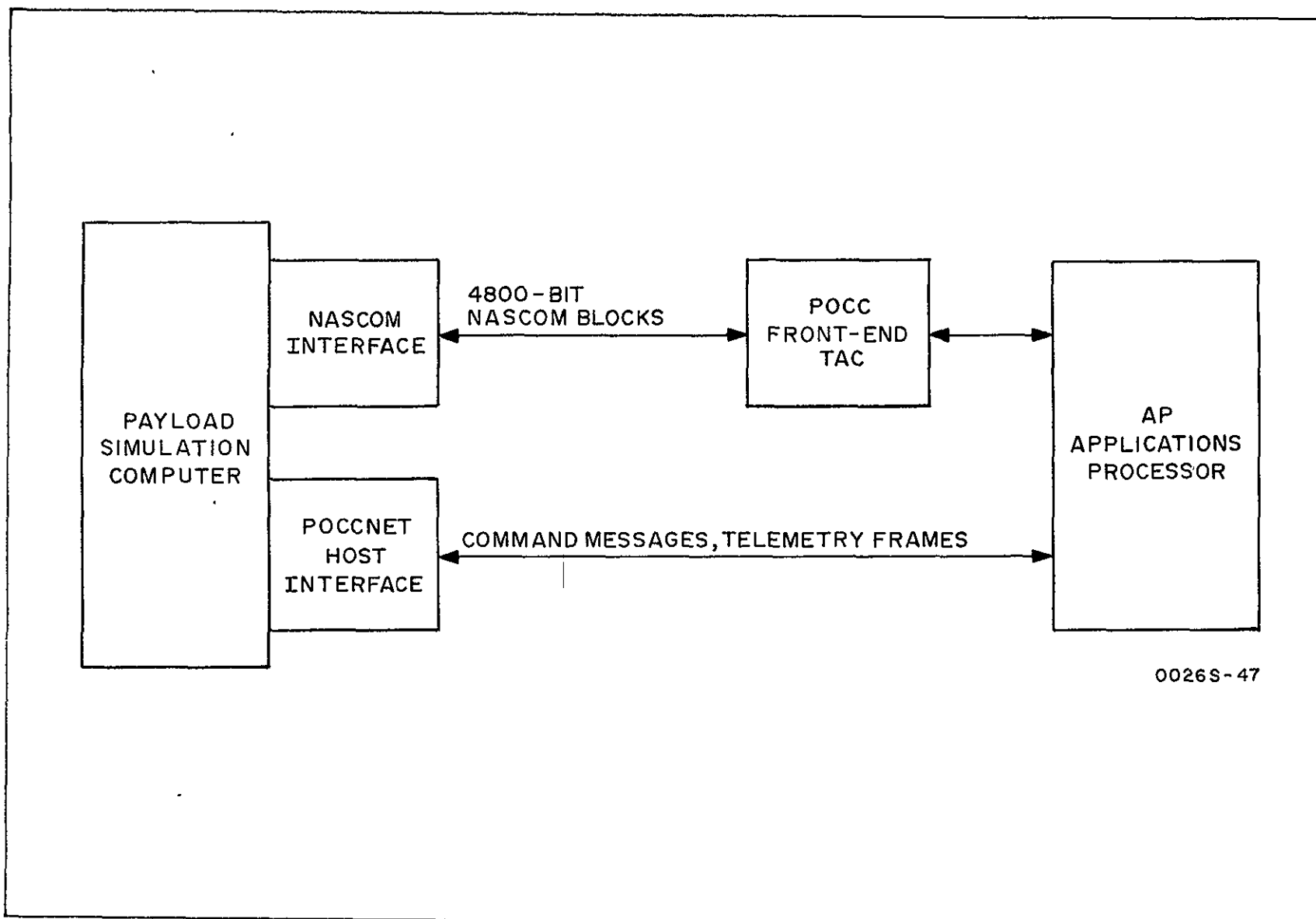


Figure 7-2. Payload Simulation/POCC Interfaces

hookup with the POCC. The lower path shows the simulation computer communicating with the AP in the POCC via the Poccnet host interface. In the "all-up" hookup, the command uplink and telemetry downlink interfaces will require no modifications to POCC software.

An alphanumeric CRT display unit will be available to the Simulation Director at the POCC to access portions of the global core data (GCD) and change values of variables of interest. This would permit modifications of input data for the various models (for example, simulation of equipment malfunction or failure and anomalies in the spacecraft subsystems, telemetry changes, and run mode selections). An identical CRT display unit will be available to the Operations Director. Also, the POCC will provide a two-way voice intercom link to the simulation facility Operations Director. The capability to monitor simulation performance via selectable formatted pages on the CRT must also be available.

7.4.2.2 Simulator Installation. The host computers presently being used as payload simulation facilities are the M&DOD IBM 360-65 and the MSC&AD PDP-11/70.

The payload simulation development efforts will utilize personnel, hardware, and software resources from these computer installations for implementation, test, integration, operations, and maintenance.

7.4.2.3 On-Line Operations. The Simulation Director will be able to control and monitor simulation operations from his console. This includes the capabilities for making changes in the simulation parameters, terminating the run, restarting with or without a new configuration, checkpointing, refreshing, and idling. These terms are explained in paragraph 7.4.3. The Operations Director will normally only monitor simulation operations from his console.

Checkpoints for continuity will automatically be created periodically while the simulator is running, so that the simulation can be restarted quickly in the event of an unexpected interruption. Manual checkpointing will also be available. The simulation will be capable of a timing fidelity close enough to the actual spacecraft to be undetectable by a human operator at the POCC.

7.4.2.4 Off-Line Operations. The simulation will be capable of running in an off-line mode for program development, testing, and analysis. The off-line mode indicates

that either the command uplink from the POCC, the telemetry downlink to the POCC, or both are disconnected. In this type of operation, simulations of the appropriate links will be available.

The simulation will be capable of producing magnetic tapes for analysis (cal comp plot tapes) or training (command history and telemetry digital tapes). An option to run the simulation in non-real-time will be available. The Operations Director's console, if used, will allow the capability to control and monitor the simulation.

7.4.2.5 Resource Sharing and Modularity. Resource sharing will be maximized within the family of payload simulations by making use of a functional module in as many family members as possible. This requires a careful analysis of the common functional elements in the various simulation types.

Reusability of the software components will be enhanced by the following:

- a. Identifying hardware subsystems to be used on one or more series of spacecraft.
- b. Carefully developing software models to parallel the hardware operation.
- c. Reflecting the modeling of subsystem installation on the spacecraft via a table rather than hard-coding it.

7.4.3 FUNCTIONAL DESIGN OF PAYLOAD SIMULATORS

To provide the capabilities necessary to meet the computational and operational requirements of the various payload simulators, the large simulation system structure of Figure 7-3 is proposed. It should be noted that it is really not a single structure because the on-line command uplink and the off-line command input may not be used simultaneously; the software OBC module and the hardware OBC may not be used simultaneously; and the off-line telemetry recording module may be used in lieu of, or in conjunction with, the on-line telemetry downlink. The various payload simulation types are achieved by selecting appropriate subsets of this simulation structure. The commonality of the various functional entities among the simulation types is evidenced in Table 7-1.

The capabilities provided by the various payload simulation functional components are described in Appendix 7B, Functional Design of Payload Simulators. The constraints, where appropriate, are also mentioned. Wherever a specific example is needed, the Multimission Modular Spacecraft (MMS) is considered.

Table 7-1
Summary of Simulation Types Versus Simulation Components and Characteristics

Simulation Components/Characteristics	Simulation Type								
	Training	Analysis	OBC Rewrite Validation	Telemetry Displays Validation	Command Validation	Experiment	Attitude Determination	Command Memory Management	Power Management
System Control									
Executive control	X	X	X	X	X	X	X	X	X
Global core data	X	X	X	X	X	X	X	X	X
Input/output processors	X	X	O		X	X		X	O
Interactive control & display	X	O	O	X	O	X	X		X
Commands									
On-line command uplink interface	X		X	X	X	X		X	X
Off-line command input	O	O		O		O			O
Command queue processor	X	O	X	X	X	X		X	X
Command front-end handler	X	O	X	X	X	X		X	X
Command history	X	O	X	X	X	O		X	O
Command processor	X	O	X			X			X
Spacecraft Models									
Actuator models	X	X	X			O			
Environment model	X	X	X			X	X		X
Spacecraft dynamics	X	X	X			O	X		
Sensor models	X	X	X			O	X		
Software OBC (Control law, CMD memory)	X	X				O		X	
Hardware OBC (Control law, CMD memory)	O		X						
Power model	X								X
Thermal model	X								
Spacecraft communications model	X								
Experiment package	X		O			X	O		
Telemetry									
Telemetry conversions package	X		X	X		X	X		
Telemetry fabrication	X		X	X	X	X	X	X	
On-line telemetry downlink interface	X		X	X	X	X	X	X	
Off-line telemetry record	O			O	O	O	O		
Ground software (POCC)									
Command uplink software	X		X		X	X		X	
Telemetry acquisition & display	X		X	X	X	X	X	X	
Attitude determination software							X		
Command memory management software								X	
Real-time capability	X	O	X	X	X	X	X	X	
Non-real-time capability		X					O		X
Legend X Component required O Component optional (blank) Component not required									

7.4.4 OPERATIONAL CAPABILITIES

Generally speaking, a payload simulation may be run in real-time or in non-real-time. A simulation run goes through three phases: initialization, simulation, and termination. The simulation may, at a given time, be in one of three possible modes: normal, idle, or step. These operational capabilities, as well as those related to checkpointing, refreshing, and interactive control facilities, are described in the following paragraphs:

7.4.4.1 Real-Time and Non-Real-Time Operation. By appropriately setting a run time parameter, the user may cause the simulation to be run in real-time or in non-real-time. The following paragraphs explain the distinction between simulation time and real-time and the meaning of real-time and non-real-time operation.

There are two independent time parameters involved in simulation computations. One of these is simply derived from the reference clock time available to the programs running on the simulation computer. This time parameter presumably grows at the same rate as an accurate wall clock or GMT time. In simulation terminology, this parameter is simply called real-time. The other independent time parameter, called simulation time, is associated with the spacecraft-related dynamic models (attitude control, experiment package, etc.). As discussed in Appendix 7B, the executive control (EXEC) program will execute the various modules on a fixed scan cycle basis. The execution of the spacecraft models on each scan causes the simulation time to be advanced by a fixed amount. The rate of growth of the simulation time then depends on how frequently the scan cycles are executed. The execution speed of the various models on the simulation computer should be such that, if scan cycles were executed without any delays between them, simulation time would grow considerably faster than real-time. Real-time operation is achieved by the EXEC program by spacing initiations of successive scan cycles such that the simulation time grows at the same rate as the real-time. During non-real-time operation, the EXEC program makes no attempt to keep simulation time in step with real-time but simply allocates CPU time to the simulation according to its priority in the system.

Real-time operation must be specified for any configuration involving the hardware OBC and/or on-line telemetry downlink to the POCC. Other configurations may be run in either real-time or non-real-time.

7.4.4.2 On-Line and Off-Line Configurations. A payload simulation will offer several configurations to the user. Basically, for the command function, the user may select either the on-line uplink from the POCC or an artificial program (such as the off-line command input routine) for introducing commands into the simulation. For the telemetry function, the user may select the on-line downlink to the POCC, a program to spool telemetry onto tape, or both of these. A configuration, then, is a specification of the way the user wants to get spacecraft commands into the simulation system and the way he wants to accept telemetry fabricated by the simulation. Obviously, the on-line uplink and the on-line downlink are mandatory when the payload simulator is to be used for the purpose of training POCC personnel. Other configurations have more specialized uses.

7.4.4.3 Phases and Modes. A simulation run passes through three phases. These are the initialization phase, the simulation phase, and the termination phase. During the simulation phase, the simulation may be in one of three possible modes. The modes are the normal mode, the idle mode, and the step mode. The idle and step modes have meaning only when the simulation is being performed while using the Interactive Control and Display Support Facility. This operational terminology of the three phases is explained in the following paragraphs.

7.4.4.3.1 Initialization Phase. During this phase, the simulation system is loaded into the computer, the various modules are initialized, and any last-minute changes to the GCD are made (by reading in name lists to override default values for control or engineering parameters, etc.). For runs which use the Interactive Control and Display Support, the system waits for user intervention. Upon user release, the on-line communications link, if required, is established. The system then exits the initialization phase and automatically enters the simulation phase.

7.4.4.3.2 Simulation Phase. This is the phase during which commands are received and responded to, telemetry is fabricated and transmitted, and spacecraft-related models are executed. The simulation may be in one of the following modes during this phase.

- a. Normal mode — In this mode, command, telemetry, and dynamic model execution activities are in full operation. This is the only mode possible for the simulation runs which do not use the Interactive Control and Display Support capability.

- b. Idle mode — This mode is entered as a result of user intervention from the CRT. In this mode, the dynamic spacecraft models program package is not executed, resulting in the simulation time becoming frozen. All other activities (command, telemetry, experiment, package execution) continue as normal. This mode is recommended for making manual checkpoints and refreshing the GCD and for starting the step mode simulations.
- c. Step mode — The step mode is suitable for a succession of user-specified simulations of limited duration. While the simulation is in the idle mode, the user enters his request for the "short" simulation run via the CRT keyboard. The EXEC program recognizes the step mode simulation request and causes the simulation to enter the step mode. During the step mode, the command, telemetry, and dynamic models execution activities are in full operation just as in the case of normal mode. At the end of the "short" simulation run, the EXEC program causes the system to reenter the idle mode.

7.4.4.3.3 Termination Phase. This phase is entered when one of the following three conditions is encountered:

- a. The specified simulation run time expires.
- b. The specified real run time expires.
- c. The user inputs a "STOP" request via the CRT keyboard.

During this phase, the simulation system is brought down in an orderly fashion. All major modules are called to complete their housekeeping, such as close out any data sets that had been opened or provide any necessary printout. The system performance summaries are printed out and the various modules are removed from the core.

7.4.4.4 Checkpointing and Refreshing. A checkpoint is a snapshot of a selected portion of the GCD onto disk. The simulation user will have the option to create checkpoints automatically during the simulation phase at fixed simulation time intervals. Optionally, checkpoints may be created during initialization and termination phases also. Manual checkpoints may be requested from the CRT at any time.

Refreshing is the act of updating or overriding a selected portion of the GCD by reading in a previously created checkpoint from disk. A refresh causes a discontinuous jump (backward or forward) in the simulation time. Normally, a refresh is made in the idle mode.

7.4.4.5 Interactive Control via Simulation Director Console. When the simulation run includes the Interactive Control and Display Support Facility, the user can perform any of the following operations to control the simulation run.

- a. Manually input changes to GCD variables before the simulation phase begins or at any other time, if appropriate.
- b. Place the simulation in the idle mode.
- c. Perform step mode "short" simulations.
- d. Create checkpoints.
- e. Refresh the GCD from a previous checkpoint and restart the simulation.
- f. Input spacecraft commands directly from the CRT.
- g. Introduce equipment malfunctions or faults into the simulation.
- h. Activate/deactivate most of the major modules.
- i. Dynamically turn on and off diagnostic printouts from the various modules.
- j. Input a new attitude for the spacecraft, effectively reorienting the spacecraft.
- k. Terminate the simulation run.

7.4.4.6 System Timing and Diagnostics. The system capabilities with respect to the run time performance of the simulation will be briefly discussed in this paragraph. It should be understood that actual implementations will vary significantly from one to another because of the complexity of the models and the capabilities of the hardware.

The EXEC program will operate on a fixed scan cycle basis, and the choice of the scan cycle time (that is, the time elapsed between starts of successive scans) will be influenced by the following considerations:

- a. Integration step size needed in the dynamics routine.
- b. The telemetry rate requirements.

- c. The maximum commanding rate.
- d. The frequency at which the control algorithm is executed in the OBC.

Whatever the choice of the scan cycle time, the system performance must meet the following requirements:

- a. To a human observer at the POCC communicating with the simulator via on-line command and telemetry, there should be no discernible difference between the spacecraft and the simulator as regards timing fidelity.
- b. The system must be able to recognize "stutters" (which occur whenever the simulation does not receive the CPU in a timely fashion) and take appropriate remedial action. There should be no long-term degradation in the operation of various model routines as a result of stuttering.
- c. Telemetry dropouts (caused by stuttering) must be held to a minimum by providing adequate pipelining of telemetry.
- d. Commands in the command queue which are ready for execution must be handled in a reasonably short period of time.

The simulation system will internally collect statistics on its own performance. (This type of activity will be controlled via user-specified options.) The performance summaries will be printed out at termination. The simulation performance monitor will have the capability of displaying run-time diagnostic messages on the CRT.

7.5 ON-BOARD COMPUTER SUPPORT

An increasingly important area of POCC responsibility is the support of space payload OBCs. Some activities required for this support are carried out on line, in the POCC AP; others may be accomplished off line. Thus, the associated applications programs may be run on the POCC AP, on the 360-65, or on the MSC&AD PDP-11/70. In the future, off-line OBC support may be provided by a Poccnet AP dedicated to this activity.

7.5.1 ON-LINE OBC SUPPORT

On-line OBC functions vary from payload to payload. In general, the POCC is responsible for loading the OBC memory, dumping the OBC memory or status buffer,

maintaining an image of the current OBC memory contents, printing OBC dumps, and controlling OBC operations. OBC memory loads may consist of programs or data blocks. OBC dumps may be taken from the entire OBC memory or may be limited to specified tables. The POCC is also usually required to compare the output of an OBC memory dump or status buffer with the current OBC core image or configuration which it maintains, displaying any discrepancies to operations personnel.

An important area of OBC support by the POCC is the preparation and uplinking of OBC data blocks. This operation usually consists of selecting the portions of a previously constructed master data set which are to be used by the OBC in the current operation period, constructing a memory image data block, and uplinking ~~this data~~ block to the OBC. Examples of the type of data which may be involved in this activity are portions of a star catalog to be used for an upcoming observing session or orbit data for the next 24 hours, selected from a data set containing orbit data for 1 to 2 weeks.

7.5.2 OFF-LINE OBC SUPPORT

The primary objective of off-line OBC support is the preparation of data and program loads for use by the POCC. A typical off-line OBC support system consists of four major components: an assembler or higher order language compiler, a relocating loader, an OBC interpretive simulator, and an executive. The assembler or compiler processes programs and data definitions for the OBC, producing absolute object modules and source listings. The relocating loader accepts the object modules produced by the assembler, resolves external references, assigns storage addresses, prints a load map, and produces an OBC memory image for uplink to the OBC. The OBC interpretive simulator takes in an OBC program load, simulates program execution, and gathers statistics on instruction use and run times. The simulator also allows tracing of program flow and selective dumping of the (simulated) OBC memory. The executive accepts control cards which specify what sort of processing is to be done and the parameter values to be used.

Off-line OBC support systems are also used to produce data sets for use by the POCC. A star catalog (of potential targets from which a subcatalog may be extracted during real-time operations) is one example of a type of data set which might be prepared off line. Another example is orbit data, which can often be generated in data sets covering a period of several weeks. A daily portion is extracted from this data set and sent to the OBC during real-time POCC operations.

7.6 COMMAND MEMORY MANAGEMENT (CMM)

The following paragraphs describe a top-level functional view of a command memory management (CMM) system, including a proposed approach for CMM on Poccnet, using the IBM 360-65 and the DBS subsystem. Appendix 7C presents a more detailed functional look at a typical CMM system, based on AEM-A.

Basically, a CMM system is designed to manage and integrate command requests involved in the creation of memory loads for an on-board memory unit referred to as a stored command processor (SCP). Frequently the SCP is implemented as a software function within the OBC. The CMM system will accept, merge, and transform command requests into a spacecraft command format from which spacecraft memory loads will be generated.

In order to accomplish its tasks, the CMM system needs to have available in its data base detailed information concerning the following:

- a. The structure and operation of the spacecraft SCP.
- b. The various command formats.
- c. Interface between the CMM facility and the POCC.
- d. Constraints and limitations imposed on the CMM system by the SCP.
- e. Any special requirements and/or considerations that may be imposed on the CMM system by the payload project.

As an example of the extent of information required, consider the data regarding the structure and operation of the spacecraft command processor which the CMM system uses. This data is functionally broken up into five categories described in the following paragraphs.

7.6.1 INTERFACE INFORMATION

The interface between the ground and the spacecraft needs to be established. This involves information about the command frame format including such items as the following:

- a. Spacecraft address.
- b. Decoder address.
- c. Operational codes.

- d. Command data-field lengths.
- e. Parity.

The complete repertoire of commands must be known as well as their types (discrete, serial, etc.).

7.6.2 MEMORY LOAD STRUCTURE INFORMATION

In this category, the following information is included:

- a. The total memory capacity of the on-board memory unit.
- b. The mix of command types allowed.
- c. The timing mechanism used for controlling the execution of commands.
- d. The maximum amount of information capable of being stored per memory location.

7.6.3 COMMAND LOADING INFORMATION

This category includes such items as the following:

- a. Categorization of commands as real-time or delayed.
- b. Mechanism for storing delayed-mode commands on-board for subsequent execution.
- c. Number of command frames needed for different types of commands (discrete, serial, etc.).
- d. Rules regarding full or partial memory loading.

7.6.4 CLOCK INFORMATION

Included in this category are the following:

- a. Size of spacecraft clock.
- b. Rules for synchronization of spacecraft clock with external timing devices.
- c. Rules for the use of the spacecraft clock in timing execution of commands.

7.6.5 OPERATIONAL MODES INFORMATION

This category includes the following:

- a. Definition of the various modes of operation of the command memory unit (normal, load, dump, etc.).
- b. Rules which indicate what controls the various modes.

SECTION 8
OPERATIONS

SECTION 8 OPERATIONS

CONTENTS

<u>Paragraph</u>		<u>Page</u>
8.1	General	8-1
8.2	Maintenance and Operations	8-1
8.2.1	Maintaining Resources Tables	8-1
8.2.2	Scheduling Resources	8-2
8.2.3	Establishing Configurations	8-2
8.2.4	Verifying Configurations	8-4
8.2.5	Maintaining Status	8-6
8.2.6	Providing Fault Isolation, Repair, and Evaluation . .	8-6
8.2.7	Providing Performance Monitor and System Analysis	8-7
8.2.8	Performing Simulations	8-9
8.2.9	Pocnet Maintenance	8-9
8.3	Data Operations Control	8-11
8.3.1	DOCC Operating Functions	8-11
8.3.2	DOCC Software Requirements	8-12
8.3.2.1	DOCC Operating System	8-12
8.3.2.2	DOCC Applications	8-14
8.3.3	DOCC Hardware Requirements	8-15
8.3.3.1	Interactive Terminals	8-15
8.3.3.2	Visual Aids	8-15
8.3.3.3	Hard-Copy Devices	8-15
8.3.3.4	Communications	8-16
8.3.3.5	Portable Test Stations	8-16
8.3.4	POCC Data Operations Control	8-16
8.4	Facilities	8-17
8.4.1	Pocnet Unique Systems	8-17
8.4.2	Space Requirements	8-19
8.4.2.1	Equipment	8-19

CONTENTS (Cont)

<u>Paragraph</u>		<u>Page</u>
8.4.2.2	Operations	8-19
8.4.2.3	Support	8-19
8.4.3	Functional Layout	8-19
8.4.4	Space Availability	8-21

ILLUSTRATIONS

<u>Figure</u>		<u>Page</u>
8-1	Formation of POCCs from Pocnet Resources	8-3
8-2	Connection Verification Over Virtual Channels	8-5
8-3	Desired Levels of Support-System Test/Verification Capabilities	8-8
8-4	Pocnet Conceptual System (Model POCC Pair, DOC/ DBS Complex, and Interfaces)	8-18
8-5	Pocnet Functional Layout	8-20

SECTION 8 OPERATIONS

8.1 GENERAL

Pocnet operations include the operation of the Pocnet system resources and the operational management of the Pocnet system. POCC operations personnel will continue to operate the POCC which, in addition to the POCC AP and peripherals, will also include the associated TAC and VIP. The Pocnet DOCC will provide the overall operational management of the Pocnet system. Briefly stated this will include the following:

- a. The support interface between Pocnet and the users. This responsibility will include all operations control functions, including preliminary, real-time, or emergency scheduling and resource allocation functions; data flow monitoring and accountability, fault isolation, repair, and evaluation; and testing and simulations involving Pocnet resources.
- b. Providing operations support in such areas as developing Pocnet schedules, documentation changes, and information requests, analyzing Pocnet service performance, the use of the DBS, and software validation and configuration control.
- c. Providing interface coordination between Pocnet and external support elements such as Orbit Determination, NASCOM, NCC, SMCC, etc.
- d. Providing the maintenance and operation of the DOCC, DBS, and Pocnet system resources.

8.2 MAINTENANCE AND OPERATIONS

A brief conceptual presentation of the functions that will need to be performed in the operation of Pocnet is provided in the following paragraphs. For additional concepts see Appendix 8A.

8.2.1 MAINTAINING RESOURCES TABLES

These tables are necessary to establish the system environment by defining the Pocnet resources and their characteristics. The tables should start with a minimum of items (that is, only the PMPs) and be expanded as each device or set of devices is moved into the allocation control of the DOCC. The tables, therefore, should

be expandable to be able to eventually contain all of the possible Poccnnet resources. Only the DOCC should be able to update the resources tables.

8.2.2 SCHEDULING RESOURCES

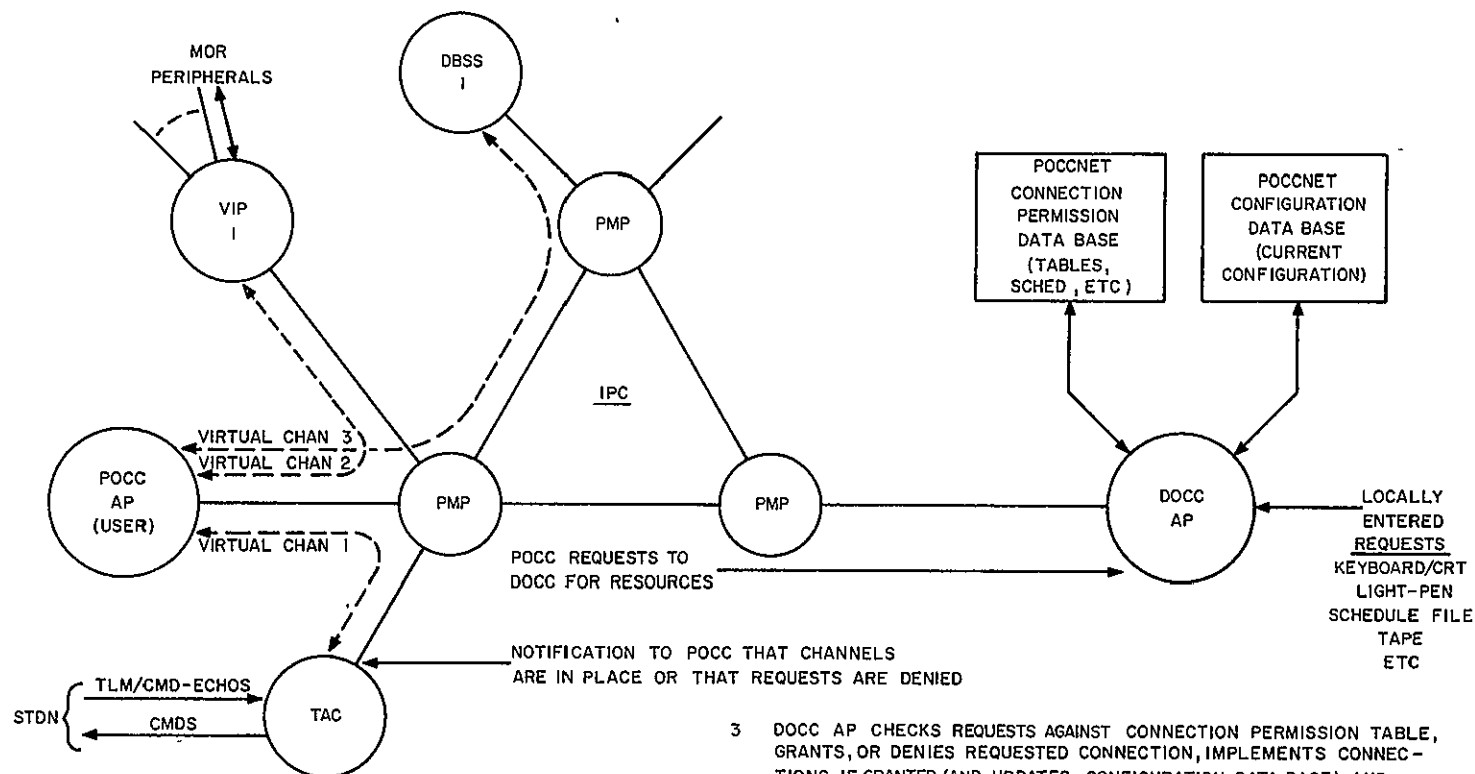
During the initial phases, Poccnnet scheduling can be performed manually by the DOCC. The requirements can be coordinated and the Poccnnet cluster can be partitioned into the necessary support configurations for the three POCC areas. The most active scheduling actions will be for the use of the resources within the MSOCC partition. A scheduling aid program should be developed, as necessary, to identify conflicts for resources within Poccnnet as well as within a partition. The scheduling system should be kept as simple as possible.

8.2.3 ESTABLISHING CONFIGURATIONS

The allocation of resources to the required configurations will need to be done on a semipermanent or dynamic basis. The configurations should be established using the Poccnnet language procedures. At first, the configurations should be so fixed that scheduling and allocating need not be dynamic (that is, set up the cluster and let it run indefinitely).

The capability should exist for both the DOCC and the DOCs to configure resources. A DOC would configure a resource by sending a configuration request to the DOCC AP where the DOCC operating system would receive the request, validate it against the resource and status tables for the allocation of the resource, validate it against the permit table for the DOC's legal connections, and issue a connection command. A DOC should have the authority (permit) to issue configuration requests for all connects and disconnects within its allocated partition of resources. (See Figure 8-1.) This will allow autonomy within a POCC, increase the transparency of the DOCC while retaining visibility of the system, and distribute the operational control function of configuring their own MORs to the DOCs.

The configuration management system should monitor the allocation times for timeouts. If a resource is connected beyond its allocated timeline, the system should issue a message to the DOCC/DOC. Only when a resource is required for a subsequent operation should the disconnect timeline be rigid. The use of the disconnects should allow for overruns of the operations whenever a resource is not



- 1 POCC REQUESTS DOCC TO CONNECT RESOURCES WITH VIRTUAL CHANNELS REQUESTS MAY BE INPUT VIA IPC OR BY LOCAL ENTRY AT THE DOCC
- 2 UPON NOTIFICATION FROM DOCC AP THAT VIRTUAL CHANNELS ARE IN PLACE, POCC AP MAY USE TEST/SENSE MSGS TO VERIFY CONNECTIONS

- 3 DOCC AP CHECKS REQUESTS AGAINST CONNECTION PERMISSION TABLE, GRANTS, OR DENIES REQUESTED CONNECTION, IMPLEMENTS CONNECTIONS IF GRANTED (AND UPDATES CONFIGURATION DATA BASE), AND NOTIFIED POCC OF DECISION
- 4 THE DURATION OF A CONNECTION MAY BE FOR MINUTES, HOURS, DAYS, ETC, AS REQUIRED
- 5 IF USER "A" DOES NOT DISCONNECT BEFORE USER "B" IS SCHEDULED TO USE THE SAME RESOURCE, THE DOCC OPERATOR WILL BE ALERTED TO RESOLVE THE CONFLICT (I.E., USER "A" WILL NOT BE AUTOMATICALLY DISCONNECTED)

POCCNET DOCC AP PARTITIONS POCCNET RESOURCES TO FORM POCCS BY CONNECTING VIRTUAL CHANNELS BETWEEN RESOURCES

0026S-49

Figure 8-1. Formation of POCCs from Pocnet Resources

allocated for another operation. As operational experience is gained on the system, the time factor can be incorporated into the scheduling requirements. The system should not automatically disconnect a resource without coordination between the DOCC and the POCC DOC.

Special configurations may be required to provide hot backup during critical portions of a mission or to distribute the processing of a mission to more than one AP.

The resources and status table information can be used to establish a permit table which should be capable of specifying that the DOC directives are valid for a specific (scheduled) time period or for an indefinite period. Only the DOCC should be able to change the permit table.

Canned procedures for configuring normal and alternate POCCs and for connecting and disconnecting various devices should be defined, using Pocnet language, and stored in the DOCC AP. The POCC APs should also store and issue canned procedures (or call them from the DOCC AP).

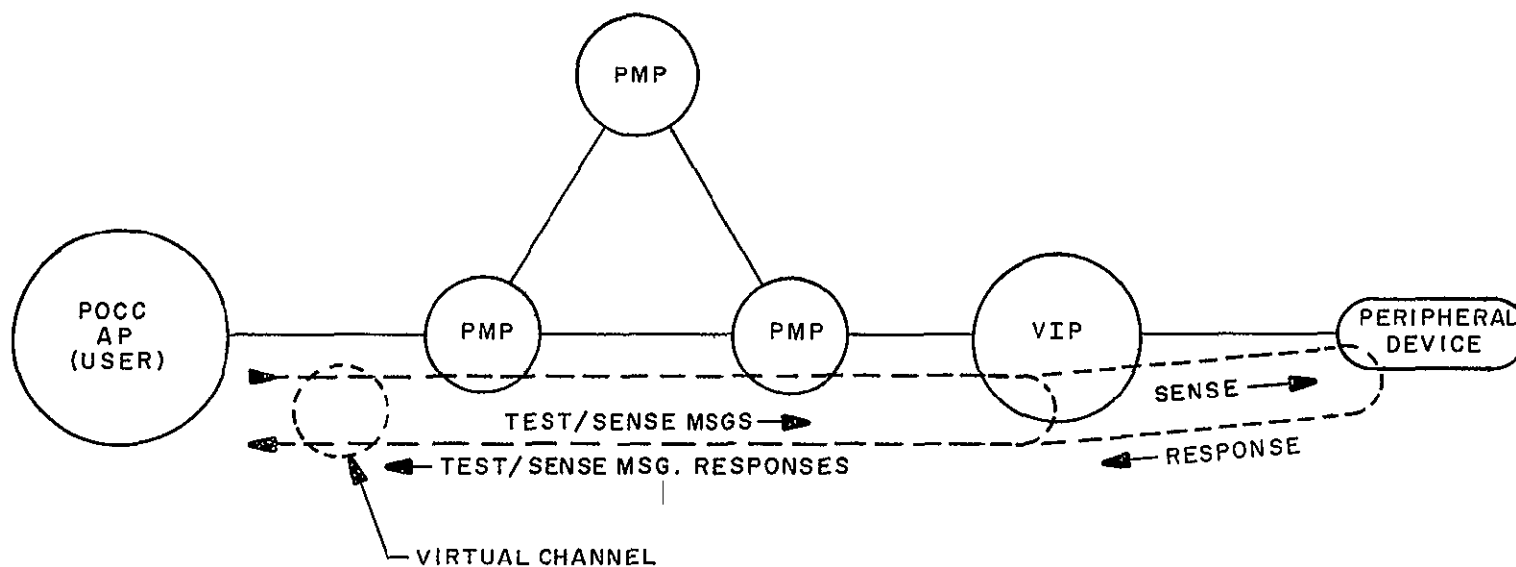
To ensure that the DOCs have the ability to configure during a DOCC AP outage, a backup method, such as a partition established in a DBS AP, should be used for this function.

8.2.4 VERIFYING CONFIGURATIONS

On a regular basis and after each configuration directive is implemented, there will be a need to verify the actual configuration.

After a configuration directive is issued, it should be followed by the transmission of a circuit (or path) assurance message to verify that it is configured. In addition, each resource of Pocnet should provide a resource assurance message after it is initialized. (See Figure 8-2.) Regular configuration checks should be made for checkpointing purposes and to compare actual configuration to scheduled configurations. The DOCC or DOCs should be able to request the configuration at any time.

Each host computer should also be able to provide its configuration at any time to the DOCC.



SENDING TEST / SENSE MESSAGES OVER VIRTUAL CHANNELS PROVIDES
POCCNET USERS AN EFFECTIVE METHOD FOR CONNECTION VERIFICATION

0026-50

Figure 8-2. Connection Verification Over Virtual Channels

8.2.5 MAINTAINING STATUS

The DOCC should be capable of monitoring and maintaining the status of the Pocnet resources. The DOCC and the DOCs should be able to query the status of Pocnet or of the POCC subset at any time. Any changes in the status of a device (hardware faults, software faults, etc.) should be reported to the DOCC as they occur. The status of the complete set of Pocnet resources should be maintained by the DOCC in a status table.

Eventually a status monitoring subsystem should be able to list any POCCs affected by a change-of-status indication by checking against the planned allocation of resources. A notification should be sent from the DOCC to the DOCs of any impending configuration problems.

8.2.6 PROVIDING FAULT ISOLATION, REPAIR, AND EVALUATION

Hardware faults, software faults, and data recipient trouble reports should be used as triggers for fault isolation procedures. If a hardware or software fault is received, the DOCC should alert the users of the data path or service to assist in determining if the fault requires immediate attention or if service should be continued in a degraded mode until after the support period is completed.

For fault-isolation purposes, the DOCC should use on-line and off-line diagnostics, alternate path tests, alternate configurations, data simulators, and alternate devices in a step-by-step mode. External data simulators, such as those available at the stations, NCC, or the Simulations Operations Center (SOC), should be called upon as needed.

The DOCC should be able to identify any and all configurations and data paths in Pocnet. The capability to display the data path for each POCC (the entry into IPC, the circuit through IPC, the exit from IPC to the MOR) is necessary so that the path can be tested to isolate any problems. For multiuser hardware, if the fault is only affecting a single user, the single user may continue in a degraded mode or be provided a replacement path (where possible). If the failure is in a device allocated to a single user, the additional option to replace the hardware should be available.

Faults within the sphere (partition) of a POCC DOC should be identified, isolated, and replaced, if possible, by the DOC. If external support is required, the DOCC should be contacted for obtaining backup resources.

To facilitate testing and evaluation, the DOCC should be able to load and initialize diagnostic software into each host remotely through the IPC. For operational redundancy, as well as for testing and evaluation, it should also be possible to load and initialize software into each host at the host and to operate it in a local mode.

An overall view of some desired levels of system testing capabilities is depicted in Figure 8-3.

The fault reporting should be established in a hierarchical manner so that each fault is reported at its highest level, not through all levels of the system. For example, if there is a failure in a VIP, it sends the fault report to the DOCC. Each peripheral device attached to the VIP does not repeat the report. Routine traffic loading reports or system performance monitor reports should not trigger the fault isolation system.

8.2.7 PROVIDING PERFORMANCE MONITOR AND SYSTEM ANALYSIS

Several methods should be used to monitor the system performance. Traffic load reports and summaries from each subsystem should be received; accumulated, and used to determine the loading the subsystems are experiencing.

Each hardware device should have a basic diagnostic that will exercise the resource to its full capacity and report the results to the DOCC upon completion. The results of these diagnostics should be accumulated on a routine basis and monitored for any degradation, and the diagnostic should be stored in the DBS for use by the DOCC or DOCs.

The summary messages and results of the Pocnet status monitoring system should also serve as inputs to the performance monitoring function.

There is an external need for a support quality evaluation report to be transmitted as a contact summary after each operation with the STDN. This data accountability report should be from the user AP or TAC. The report should also be sent to a Data Accountability System (DAS) and the DOCC.

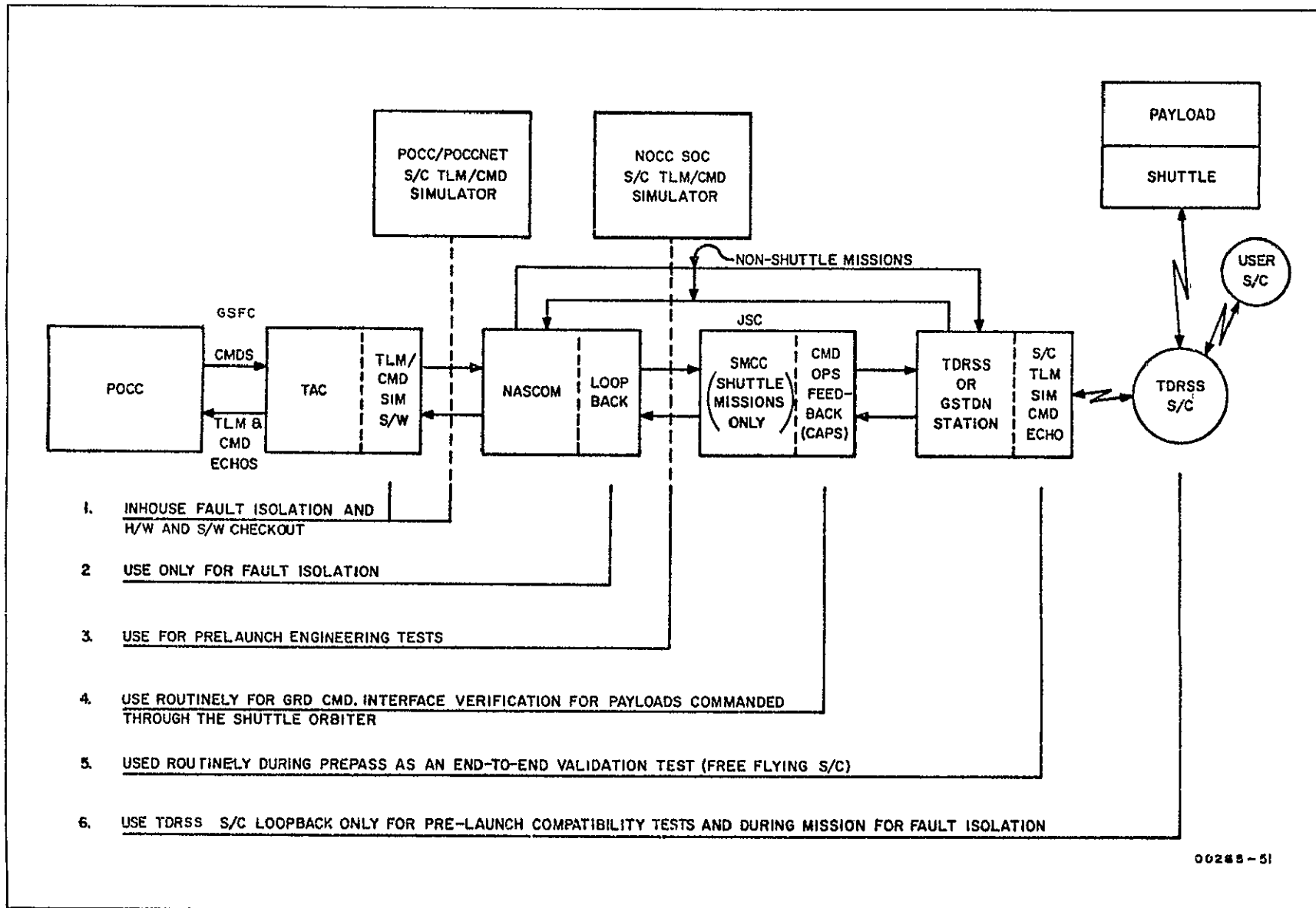


Figure 8-3. Desired Levels of Support-System Test/Verification Capabilities

8.2.8 PERFORMING SIMULATIONS

The DOCC should make use of various simulators and simulation techniques in the operation of the Poccnet system. Some of the capabilities that should be provided are described as follows:

- a. The capability to connect a spacecraft telemetry and command simulator to a TAC so that it appears to the POCC as if the data is being received from the STDN (TDRSS or GSTDN). The POCC AP should be able to communicate with the simulator and control it as if it were a spacecraft. A TAC should be able to provide a fixed-format telemetry input for simulation purposes and should be capable of returning a command echo to the POCC.
- b. Data simulators that can be connected through the Poccnet system and used in conjunction with the traffic loading/performance monitoring system to perform system loading studies, alternate configuration loading, overload thresholds, etc.
- c. The ability to connect to available simulation capability external to Poccnet. Examples are the portable spacecraft simulators at the stations or the SOC, the station simulator at the SOC, and simulators operated by the NCC.

8.2.9 POCCNET MAINTENANCE

Poccnet maintenance will follow through several levels of fault identification and diagnostics. Faults will be detected either through the status monitoring system or from a fault indication. Once a fault is detected, the DOCC will verify its effect on the users. If the fault is not catastrophic, the users may prefer to continue in a degraded mode to complete an operation in progress.

Using fault isolation procedures and tools, the DOCC/DOC will isolate the fault to a particular resource and load and run an operational level diagnostic to further identify the fault. The immediate problem will be solved by either a software command or by physically replacing the resource. At this point, operations will turn the resource over to maintenance for any full diagnostic testing and repair. A device that cannot be replaced will be allocated to corrective maintenance in the status table.

When a resource is returned from maintenance, it will be reinstalled, deallocated, from maintenance, evaluated for operational use with an operational level diagnostic, and returned to operational status.

Some diagnostic testing and maintenance of the devices may be distributed along with the resources; for example, if a TAC is uniquely tailored for use by a certain POCC and is located with the AP for that POCC, maintenance for that TAC can be distributed to that POCC.

To aid in testing and evaluation of problems, it should be possible to load, initialize, and operate each host locally, as well as being able to accomplish these functions from the DOCC via the IPC.

The system planning and implementation should include the following items that will improve the maintainability of the Pocccnet resources:

- a. Each resource should have its own independent dc power source.
- b. The hardware should be collocated where possible so that using a spare as a replacement becomes more useful. The PMPs should be collocated with the exception of those that are dispersed to remote members of the network. The gateways should be collocated with the PMPs.
- c. The resources should be designed and configured to permit fast physical replacement as an option.
- d. The ac power sources for the resources should be distributed to preclude catastrophic power loss.
- e. The spare TACs should have data lines to NASCOM to aid in fault isolation, repair, replacement, or alternate uses such as backup support.
- f. VIPs and TACS should be collocated with the normally allocated AP.
- g. To reduce the complexity of the software involved for each VIP and to reduce the effort required in coordinating and performing preventive maintenance, remedial maintenance, software testing, and reconfiguring, it would be best to have each VIP allocated to only one MOR. This situation, however, could create a single-point failure for a MOR. One possible solution would be to configure VIPs to support particular MORs and to

provide one or two connections to an alternate MOR. The MORs could have the connections on the alternate VIPs as backup in case of VIP failure. To increase the flexibility of this option, the connections within a MOR should be switchable among similar devices. The MOR could then switch the desired devices to the backup VIP connections.

8.3 DATA OPERATIONS CONTROL

Data operations control encompasses the functions and tools necessary to maintain and operate Pocnet. This includes the function of the DOCC, its hardware and software, and the associated functions of the DOCs.

8.3.1 DOCC OPERATING FUNCTIONS

The following are the operating functions of the DOCC:

- a. Maintain the Pocnet resources tables. This is a reasonably static interactive operation for table maintenance.
- b. Provide Pocnet scheduling. Coordinate scheduling, resolve conflicts, manually set up a schedule, and update the schedule as necessary. Eventually this will include operating scheduling aid programs.
- c. Operate the Pocnet status monitoring systems (SMS). This operation is semiautomatic, requires operator interactive functions when a status goes red, and could stimulate schedule changes, configuration changes, and followup actions by the operator.
- d. Operate Pocnet configuration management system (CMS). The DOCC operating system (DOS) will use automatic sequence processing (extension of STOL) to do most of the configuration management by monitoring DOC requests. For nonscheduled configurations, fault recovery, and DOCC configuration procedures, the operator will be required to interact with the system. The DOCC operators must also coordinate and communicate with DOC operators for configuration changes, conflicts, problems, etc.
- e. Operate fault isolation system. When a fault is detected, identify and isolate the fault and, if possible, replace the resource or provide alternate configurations to keep the support going. If necessary, replace

device and turn it over to maintenance. Operationally evaluate a device after its return from maintenance and reinstall in Poccnet.

- f. Operate DOCC AP and peripherals, including DOCC operator positions and associated devices/tools.
- g. Provide O&M documentation for DOCC and DBS. (Users Guide, Operations Control Documents, Interface Control Documents, etc.)
- h. Provide required system reports and analysis.
- i. Coordinate and manage interfaces with external users. (ITDS, NCC, SMCC, etc.)
- j. Provide maintenance support to POCCs on Poccnet devices.
- k. Participate in mission readiness tests for new POCC/DOC interfaces and configurations.
- l. Provide mission planning aids, information, etc., for use by all POCCs.
- m. Operate an interactive terminal device for input/output functions with NCC scheduling system.

8.3.2 DOCC SOFTWARE REQUIREMENTS

To perform the operating functions outlined will require a DOCC operating system (DOS) and a number of DOCC applications programs. The functional descriptions of the software requirements are outlined in the following paragraphs. The operating system, the status monitoring system, and the configuration management system will have to operate simultaneously.

8.3.2.1 DOCC Operating System. The DOS will be the system that manages the DOCC AP and its functions in support of the DOCC in the operation of Poccnet. Some of the features required of the DOS are as follows:

- a. Initialization — The DOS must be capable of initializing the AP. This includes the creation and loading of the proper data sets and software. The system should be capable of either a cold start or a warm start. The warm start recovery capability should be able to read any checkpointed data sets for use in the restart. Restarts will require the reading of

status of all system resources and of the configuration of all resources. The restart capability must not disrupt the configurations and operations in progress. If the DOCC AP fails, it should not disrupt the Pocnet configuration in effect. A warm-start backup system should be stored in DBS and be ready for use on operator command.

- b. Checkpointing — The necessary operational data sets must be checkpointed to backup data sets on the DOCC AP or the DBS for use in recovery operations and for performance analysis.
- c. Priority — The system must have a priority scheme for handling the operating system and applications programs in a workable manner.
- d. File management — The system must provide file management services (access, protection, etc.) for the operational files and programs. The operational files should reside on the DOCC AP with dual redundant backup files on the DBS.
- e. Language interface — The system must be able to interface with the Pocnet STOL.
- f. Operator interface — The system must provide an interactive interface to the DOCC operator for providing functional control of the system and for DOCC/DOC communications, DOCC/Pocnet communications, and keyboard, light pen, and CRT processors.

As an extension to STOL, there should be a generalized interactive control function to call and operate the following ancillary functions used by the operators:

- a. To call diagnostics.
- b. Perform simulations.
- c. Communicate with DBS.
- d. Communicate with DOCs.
- e. Communicate with external users.
- f. Load software into Pocnet devices.
- g. Load and store canned procedures.

8.3.2.2 DOCC Applications. The DOCC applications programs required to perform the maintenance and operations of Pocnet in conjunction with the DOS are as follows:

- a. Resource data base management — A program to provide the capability to create the resource data base tables and to interactively maintain the contents of the tables.
- b. Status monitoring system — The status monitoring system software must be capable of determining and maintaining the status of the Pocnet system.
- c. Configuration management system — The configuration management system must be capable of establishing and verifying the configurations within the Pocnet system.
- d. Performance monitoring and system analysis system — The performance monitoring system software should provide the capability to monitor the performance of the system or configuration by utilizing reports generated by the hosts and PMPs, the APs, the status monitoring system, or any diagnostics operated for performance tests.
- e. Fault isolation, repair, and evaluation system — The fault isolation, repair, and evaluation system should be capable of assisting the DOCC operator in isolating a fault condition to the lowest possible level and assist in preparing procedures to bypass the fault area if possible.
- f. Display generation — There should be a generalized capability to generate a number of interactive information displays for use by the DOCC and DOC operators.
- g. Additional resource usage — The Pocnet software system should also be developed with the planned later addition of other applications that are not part of the original system but are anticipated to become Pocnet residents in the future. Systems/programs in this category are a mission planning system and a NCC scheduling system interface.
- h. In addition to the DOCC operational requirements, there should be a capability to downline-load bootstrap programs into host computers to enable the hosts to load software programs from the DBS or from a local peripheral.

- 1 For operational use, it should be possible for the DOCC to load, initialize, and operate each host via the IPC.
- j. Software should be available to process output messages and convert them into standard teletype format for transmission via NASCOM.

8.3.3 DOCC HARDWARE REQUIREMENTS

The DOCC hardware required is described in the following paragraphs.

8.3.3.1 Interactive Terminals. The DOCC operators will require the use of interactive computer terminals to maintain and operate the Pocnet system. Each terminal should contain a color CRT display, keyboard panel, and light-pen capability. Each terminal should be capable of displaying dynamic graphic displays, tabular alphanumeric displays, and combinations of the two.

The interaction via the keyboard, light pen, and CRT will be the prime method of calling and operating the various applications functions of the DOCC software system and of monitoring the operation and status of the Pocnet system.

Computer-driven information displays will be the primary method of presenting information to the operators. However, to ensure that operator attention is obtained when required, audio/visual alerts/displays, in addition to the CRT display, should be provided.

The interactive terminals will also be used to perform ancillary functions such as performance analysis, data base maintenance and review, and loading of simulation or host computer software from the DBS.

To prevent single-point failure and to perform all of the functions, there should be at least three of these terminals in the DOCC.

8.3.3.2 Visual Aids. Light-emitting diode (LED) monitors should be provided on the input lines as operational aids. CRT scope monitors should be provided along with the capability to patch the monitors to the inputs for operational monitoring and fault isolation.

8.3.3.3 Hard-Copy Devices. CRT hard-copy (video printers) and line printer output devices will be required for DOCC operations. The CRT video printers should be located in the DOCC. The line printer capability can be provided via the

DOCC AP or the DBS as long as it is readily accessible. Two video printers should be installed, which will also be usable as quiet printer plotter peripherals on the DOCC AP.

8.3.3.4 Communications. The DOCC operators will require voice communications to the POCC DOCs and all equipment areas of the Poccnet system (DBS, IPC area, DOCC AP, etc.). For general communications use, the DOCC should have send and receive teletype capability. In order to disseminate information to the DOCs, MORs, and other operations areas, the DOCC should have access to the closed-circuit television system.

8.3.3.5 Portable Test Stations. To facilitate testing of the scattered host computer devices (PMPs, VIPs, TACs), a portable test station should be implemented for each physical area. This test station should be able to connect to any host for testing, fault isolation, display of data blocks, etc.

8.3.4 POCC DATA OPERATIONS CONTROL

POCCs will continue to have their own DOCs and data operations control areas and equipment for performing the type of functions they perform today, plus additional functions required to utilize Poccnet. For example, a POCC DOC will perform the following:

- a. Configure the POCC computers and initiate request directives to configure (connect and disconnect) Poccnet resources required by the POCC.
- b. Direct the POCC computer operators and data clerks.
- c. Establish voice/data interfaces with areas outside of the POCC (for example, GSTDN and TDRSS, attitude determination, experimenters, command memory management (CMM), Poccnet DOCC, etc.).
- d. Maintain logs and records of operations, as required.
- e. Record and report support deficiencies.
- f. Assist in isolating and providing or coordinating corrective action for equipment failures.
- g. Serve as the interface between the POCC MOR personnel (spacecraft controllers/analysts, etc.), the NCC, and the DOCC for obtaining STDN and Poccnet support.

- h. Monitor the status and performance of POCC and POCC-used Pocnet resources.
- i. Operate interactive terminal devices for input/output functions with the NCC scheduling system.
- j. Perform routine end-to-end system configuration verification tests (verify that POCC and Pocnet resources are properly configured and operational and flow test data between the POCC and the STDN, CMM, support computing, spacecraft simulator, experimenter facilities, etc.).
- k. Coordinate and monitor the transmission of data between the POCC, Pocnet, and other facilities.

8.4 FACILITIES

Providing a facility for any new computing or operations activity entails two basic considerations: the space required and the utilities. Since utilities requirements, for both power and air-conditioning, are dependent upon rather specific descriptions of hardware to be purchased and since these systems are in only the conceptual stages of planning, they will not be addressed at this time.

Consideration is given to defining the requirements for plant space to accommodate that portion of the Pocnet system which is unique to Pocnet and excluding consideration for new or different requirements for individual control centers which may be a part of that system.

8.4.1 POCCNET UNIQUE SYSTEMS

The unique systems (those which will not be an integral part of some existing POCC facility and which will require new space) are those shown conceptually in the DOC/DBS complex of Figure 8-4.

The complex is considered to include three APs of the PDP-11/70 size, two VIP computers such as the PDP-11/34, several one-rack minicomputers to serve as IPC and gateways, and several additional racks of sharable peripherals and jointly accessible tape units and disks.

In addition, a 3- to 4-position console for operations activity will be included, and space must be provided for such support functions as systems engineering and programming, user interface activities, and Pocnet maintenance and operation administration.

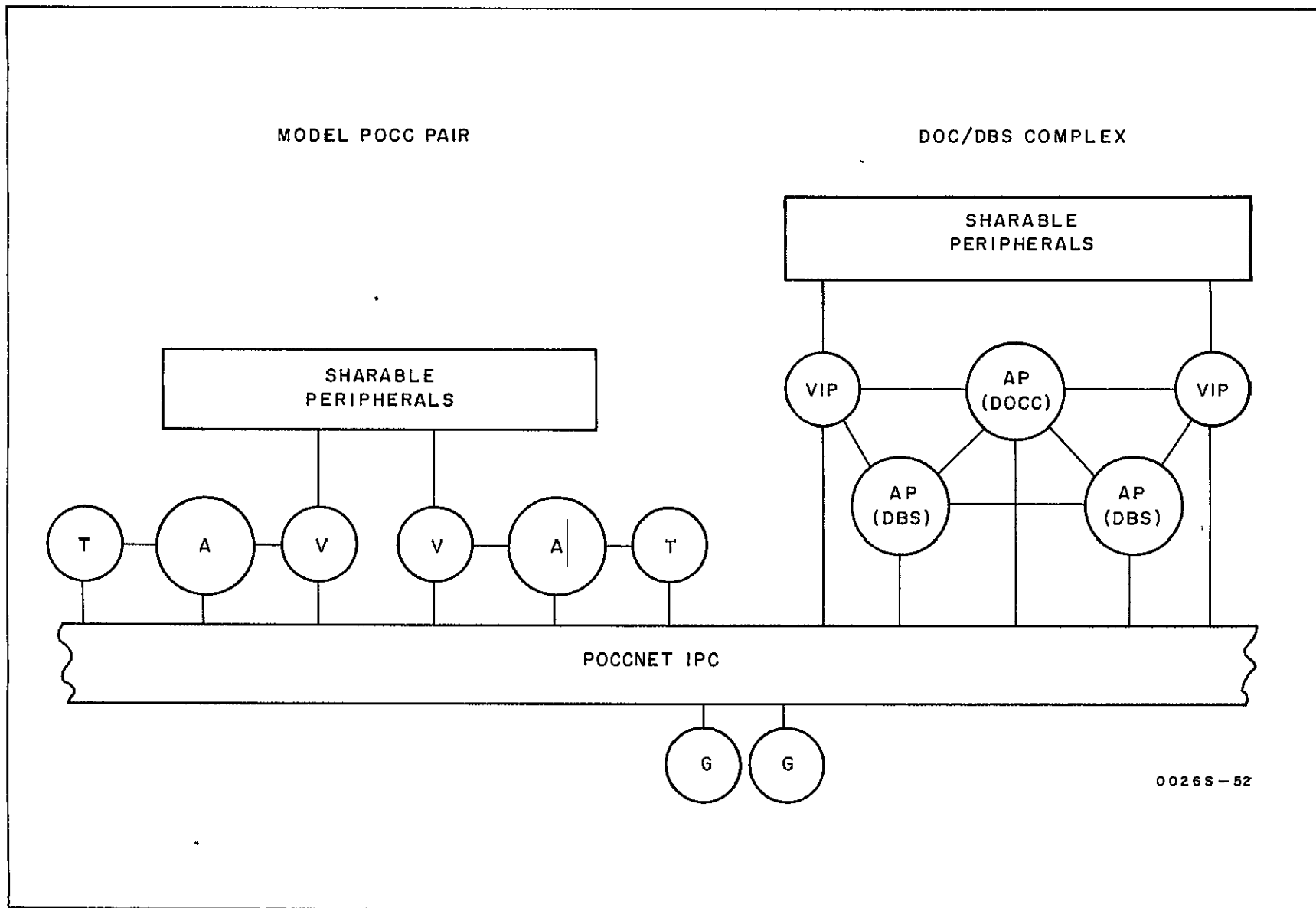


Figure 8-4. Pocccnet Conceptual System (Model POCC Pair, DOC/DBS Complex, and Interfaces)

8.4.2 SPACE REQUIREMENTS

Space can be classified as one of three basic types (equipment, operations, or support) according to its function and its utilities requirements. Equipment space has the most rigid requirements for plant resources and usually requires false floors. Operations areas are less rigid but have similar requirements for utilities and their locations must be in the immediate proximity of the equipment they control. Support areas vary in their requirements depending on function but usually do not require the same level of plant resources or false floor, and much of the support area does not need to be in the same immediate vicinity of the other areas.

8.4.2.1 Equipment. The total hardware system will consist of approximately 67 racks of computer and peripheral devices, each rack assumed to be 24 by 30 inches and requiring a minimum of 24 inches of access on two sides. Total area required: 1750 square feet.

8.4.2.2 Operations. Operations control monitoring and resource allocation will be conducted from a DOCC area consisting of approximately eight console-type racks with CRTs and communications. It is planned for this area to be large enough to include on-shift administrative and clerical activities for the total facility. Total area required: 240 square feet.

8.4.2.3 Support. There will be three support functions performed within Pocnet: systems support (both programming and engineering), Pocnet M&O administration, and user interface (mostly for data base storage). These each may be allocated discrete areas within or near the Pocnet primary facility. Total engineering area: 365 square feet; total M&O administration area: 600 square feet; total user interface area: 340 square feet. Total support area required: 1305 square feet.

Therefore, the grand total requirement for Pocnet is the sum of these areas or approximately 3300 square feet, nearly 2000 of which must be located on false floor with full utility services for equipment classified space.

8.4.3 FUNCTIONAL LAYOUT

Figure 8-5 is a functional layout of the Pocnet facility showing the types of areas required and a typical equipment arrangement. There are many variations to both

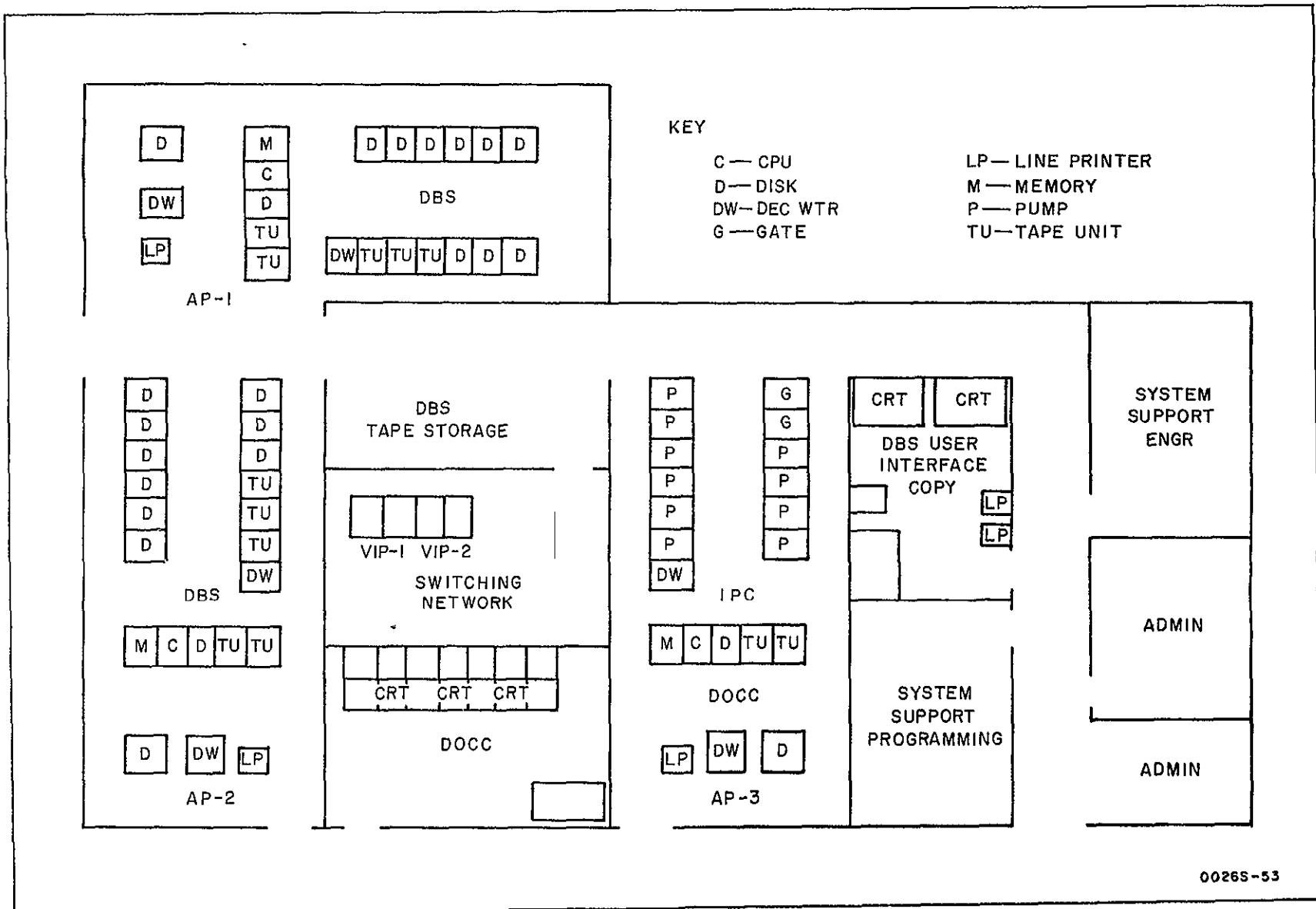


Figure 8-5. Pocnet Functional Layout

space layout and equipment arrangements and neither will be finalized until a more exact identification of the equipment is made and until a location is selected for implementing the new facility.

8.4.4 SPACE AVAILABILITY

The requirement is for 2000 square feet of false floor area and another 1300 square feet of support area in or near that space for a total of 3300 square feet.

At the present time, projected space utilization for buildings 3/14 shows that the first control center (or other large-size computer complex with the type and amounts of space required for Poccnet) will become available in September 1979 when the budgeted support for ATS terminates. This, of course, assumes no extensions beyond current guidelines, although that possibility exists.

SECTION 9
USING POCCNET

SECTION 9
USING POCCNET

CONTENTS

<u>Paragraph</u>		<u>Page</u>
9.1	General	9-1
9.2	Information	9-1
9.2.1	Pocchnet User Handbook	9-1
9.2.2	Pocchnet User Interface Engineer	9-2
9.3	Formal Requirements	9-2
9.3.1	Configuration Control Board	9-2
9.3.2	Interface Control Documents	9-3
9.4	User Operations Interface	9-3

SECTION 9 USING POCCNET

9.1 GENERAL

A Poccnet user is simply defined as any system which either inputs data into or receives data from Poccnet. In general, users of Poccnet will find their interfaces simpler but more flexible than previous POCC interfaces. This is due to the inherent characteristics of a distributed network and the implementation of standard solutions to common requirements. The Poccnet methods of providing interface solutions are described in the following paragraphs.

Users interface with Poccnet at different functional levels roughly corresponding to the time before launch and the phase of the user system development process. However, the boundaries of these levels are not rigidly defined, and, in fact, a user may be involved at several levels simultaneously. These levels are explained in the following paragraphs.

9.2 INFORMATION

Two official sources of information will be established to help inform the user: the Poccnet User Handbook and the Poccnet interface engineer.

9.2.1 POCCNET USER HANDBOOK

A Poccnet User Handbook will provide the user with most of the information required to interface with Poccnet. The handbook will also reference technical manuals where more detailed interface information may be obtained. The contents of the handbook will include the following.

- a. Placing requirements on Poccnet — The user will be informed as to how he may place formal requirements on Poccnet. Requirements will be stated in the areas of software, hardware, integration, testing, facilities, data base storage, etc.
- b. Standard systems and services — This section will describe the standard systems and services available to the users, including data base management, displays and display languages, command management, telemetry processing, external (to the POCC) system interfaces, etc.

- c. Operational procedures — The operational procedures will be defined to the user in terms of how to use the system, how to communicate with other ground system entities, and how to interpret displayed system information.
- d. Software engineering standards — It is anticipated that users will develop unique software either for running on Poccnet systems or for communicating with Poccnet. Users will be required to adhere to the Poccnet Software Engineering Standards. These standards will include the STOL and system protocols.
- e. Tests and simulations — The user will be informed as to the tests and simulations required to integrate his unique hardware and software systems into Poccnet and to proof-test the total system.
- f. Troubleshooting procedures — When problems surface during prelaunch testing or during actual operations, procedures will be defined to assist in the isolation and correction of the problems.
- g. Configuration Control Board — A Configuration Control Board (CCB) will be established to control the configuration of all systems (hardware and software) residing within Poccnet or acting as a user of Poccnet.

9.2.2 POCCNET USER INTERFACE ENGINEER

An individual will be officially assigned to serve as the interface between the user and Poccnet. The interface engineer will provide the user with information about Poccnet and will assist the user in meeting his support needs.

9.3 FORMAL REQUIREMENTS

Once a user is familiar with Poccnet and knows what he wants from Poccnet, he then is prepared to place formal requirements upon Poccnet. The following paragraphs describe the formal mechanisms for accomplishing his goals.

9.3.1 CONFIGURATION CONTROL BOARD

A Configuration Control Board (CCB) will be constituted with the authority to approve all requirements on Poccnet systems and any changes to those systems. Since Poccnet has incremental expandability as one of its fundamental drivers, the evolution of Poccnet systems is expected to be much simpler than former POCC

systems. The CCB will provide the rational configuration management to prevent the proliferation of unique systems and the implementation of systems not conforming to Poccnnet standards.

9.3.2 INTERFACE CONTROL DOCUMENTS

All Poccnnet interfaces will be managed and controlled by Interface Control Documents (ICDs). The ICD will spell out the letter and intent of each interface agreement. ICDs will spell out communications protocol, data rates, data formats, electrical and mechanical interfaces, etc. No interface will be allowed with Poccnnet which is not controlled by an ICD.

9.4 USER OPERATIONS INTERFACE

The users operations interface with Poccnnet will be with the Poccnnet DOCC. The DOCC will provide operations support in developing Poccnnet schedules, documentation, DBS use, obtaining information, and establishing configurations. The DOCC will provide interface coordination with external support elements and will provide assistance in planning mission readiness tests, simulations, or in obtaining special support for critical support periods.

In the event of problems, the DOCC will assist in fault isolation, device or path replacement, and obtaining alternate devices or processes.

LIST OF ABBREVIATIONS AND ACRONYMS

LIST OF ABBREVIATIONS AND ACRONYMS

ac	alternating current
ACS	attitude control subsystem
ADCCP	advanced data communication control procedures
AE	Atmosphere Explorer
AEM	Applications Explorer Mission
ALPHA	alpha cluster
AMPS	atmosphere, magnetosphere, and plasmas in space
AOS	acquisition of signal
AP	applications processor
ASP	automatic sequence processor, astronomy spacelab payload
Atrex	Astrophysical Transient Explorer
ATS	Applications Technology Satellite
ATSOCC	ATS Operations Control Center
BER	bit error rate
C&DH	communications and data handling
CATV	cable television
CCB	Configuration Control Board
CITE	cargo interface test equipment
CMD	command
CMM	Command Memory Management
CMS	Command Management System
CPU	central processor unit
CRT	cathode ray tube
CSC	Computer Sciences Corporation
CSSG	Common Software Steering Group
DAS	Data Accountability System
DBMS	data base management system
DBS	data base storage
dc	direct current
DDL	data definition language
DEMOS	distributed environment for mission operations support
DMA	direct memory access

DOC	data operations control
DOCC	Data Operation Control Center
DOCIL	data operations control interactive language
DOD	Department of Defense
Domsat	Domestic Satellite
DOS	DOCC operating system
EVAL	Earth Viewing Applications Laboratory
FES	fine error sensor
GCD	global core data
GPC	general-purpose computer (on Orbiter)
GPS	global positioning system
GSTDN	Ground Spaceflight Tracking and Data Network
HDLC	high-level data link control
HOL	high-order language
H/W	hardware
IGS	inertial guidance system
I/O	input/output
IPC	Inter-Process Communication
IPF	Image Processing Facility, IUS Processing Facility
ITDS	Integrated Telecommunications Distribution System
IUEOCC	International Ultraviolet Explorer Operations Control Center
JSC	Johnson Space Center
kbps	kilobits per second
KSA	Ku-band SA
KSC	Kennedy Space Center
Landsat	Land Satellite
LCC	Launch Control Center
LCI	logical channel identifier
LOS	loss of signal
LP	line printer
LPS	launch processing system
MA	multiple-access TDRSS service
M&DOD	Mission and Data Operations Directorate

Mbps	megabits per second
MCR	Mission Control Room
MDHS	Meteorological Data Handling System
MDM	multiplex/demultiplex
MILA	Merritt Island launch area
MMS	Multimission Modular Spacecraft
MOC	Mission Operations Center
MOR	Mission Operations Room
MPS	mission planning system
MSC&AD	Mission Support Computing and Analysis Division
MSOCC	Multi-Satellite Operations Control Center
NASCOM	NASA Communications Network
NCC	Network Control Center
O&C	operations and control
O&M	operations and maintenance
OAO	Orbiting Astronomical Observatory
OBC	on-board computer
OCC	Operations Control Center
OI	orbiter operational instrumentation
OPF	Orbiter Processing Facility
OSCF	Operational Support Computing Facility
OSO	Orbiting Solar Observatory
P	pitch (axis)
PC	physical channel
PCM	pulse code modulation
PCR	payload changeout room
PFTP	Pocnet file transfer protocol
PGS	payload ground station
PMP	Pocnet message processor
POCC	Project Operations Control Center
Pocnet	POCC Network
PPF	Payload Processing Facility
QA	quality assurance
R	roll (axis)

RABD	random-access block display
RIC	Remote Information Center
SA	single-access TDRSS service
SAS	Small Astronomy Satellite
S/C	spacecraft
SCADI	Serial Communications and Display Interface
SCE	spacecraft command encoder
SCR	strip-chart recorder
SDPF	Spacelab Data Processing Facility
SEM	Software Engineering Manager
SEP	Software Engineering Panel
SMCC	Shuttle Mission Control Center
SOC	Simulations Operations Center
SSA	S-band SA
STDN	Spacecraft Tracking and Data Network
STOL	systems test and operation language
STS	Space Transportation System
S/W	software
TAC	telemetry and command processor
TDRS	Tracking and Data Relay Satellite
TDRSS	Tracking and Data Relay Satellite System
Telops	telemetry on-line processing system
TLM	telemetry
TTY	teletype
VD	virtual device
V/G	VIP or gateway host computer
VIP	virtual interface processor
VT	virtual terminal
VU	virtual user

BIBLIOGRAPHIC DATA SHEET

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.	
4. Title and Subtitle PAYLOAD OPERATIONS CONTROL CENTER NETWORK (POCCNET) SYSTEMS DEFINITION PHASE STUDY REPORT		5. Report Date January 1978	
		6. Performing Organization Code	
7. Author(s)		8. Performing Organization Report No. MOD-6SR/0178	
9. Performing Organization Name and Address Goddard Space Flight Center Greenbelt, Maryland		10. Work Unit No.	
		11. Contract or Grant No.	
12. Sponsoring Agency Name and Address Goddard Space Flight Center Greenbelt, Maryland Richard desJardins, Study Manager		13. Type of Report and Period Covered Preprint (TM)	
		14. Sponsoring Agency Code	
15. Supplementary Notes Prepared with the support of Computer Sciences Corporation, Westinghouse Electric Corporation and OAO Corporation, under various NASA contracts. <i>L.A.</i> <i>P. #5.</i>			
16. Abstract. This report presents the results of the studies performed during the systems definition phase of Poccnet, an interconnected system of standard Payload Operations Control Centers (POCCs) for GSFC for the 1980s. Work is still being done in a number of study areas; in these cases, the conclusions to the time of this writing are presented. This report describes the Poccnet concept and also includes an analysis of system requirements and an evaluation of alternative systems concepts. Preliminary designs of some subsystems are presented. In addition, various methods for development of highly reliable, reusable software are evaluated, and a recommended software engineering standard approach is given. A number of POCC application areas, such as command management, on-board computer (OBC) support, and simulation, were also studied. Other areas of investigation included the operation of Poccnet systems, the facility requirements for Poccnet, and using Poccnet.			
17. Key Words (Selected by Author(s)) computer network, command and control, distributed computing		18. Distribution Statement	
19. Security Classif (of this report) Unclassified	20. Security Classif (of this page) Unclassified	21. No. of Pages 285	22. Price*

INSTRUCTIONS FOR COMPLETING THE BIBLIOGRAPHIC DATA SHEET

(Refer to GMI 2220 5A, Approval and Special Requirements for GSFC Publications)

Make items 1, 4, 5, 9, and 13 agree with the corresponding information on the report cover. Use all capital letters for title (Item 4). Leave items 2, 6, 10, 14, and 18 blank. Complete the remaining items as follows:

- 3 Recipient's Catalog No. Reserved for use by report recipients.
- 7 Author(s). Include correspondence information from the report cover. In addition, list the affiliation of an author if it differs from that of the performing organization.
- 8 Performing Organization Report No. Insert if performing organization wishes to assign this number.
11. Insert the number of the contract or grant under which the report was prepared.
- 12 In addition to the name and address as given on the cover, include the name of the GSFC technical monitor for whom the work was performed.
15. Supplementary Notes. Enter information not included elsewhere but useful, such as: Prepared in cooperation with. . . Translation of (or by). . . Presented at conference of. . . To be published in.
- 16 Abstract. Include a brief (200 words) factual summary of the most significant information contained in the report. If possible, the abstract of a classified report should be unclassified. If the report contains a significant bibliography or literature survey, mention it here.
17. Key Words. Insert terms or short phrases selected by the author that identify the principal subjects covered in the report, and that are sufficiently specific and precise to be used for cataloging.
- 19 Security Classification (of report). NOTE. Reports carrying a security classification will require additional markings giving security and downgrading information as specified by the Security Requirements Checklist and the DoD Industrial Security Manual (DoD 5220.22-M.)
20. Security Classification (of this page). NOTE: Because this page may be used in preparing announcements, bibliographies, and data banks, it should be unclassified if possible. If a classification is required, indicate separately the classification of the title and the abstract by following these items with either "(U)" for unclassified, or "(C)" or "(S)" as applicable for classified items.
21. No. of Pages. Insert the number of pages.
22. Price. Insert the price set by the Clearinghouse for Federal Scientific and Technical Information or the Government Printing Office, if known.